Pattern Recognition Letters 32 (2011) 1089–1099

Contents lists available at ScienceDirect

Pattern Recognition Letters

journal homepage: www.elsevier.com/locate/patrec

Offline handwritten Amharic word recognition

Yaregal Assabie [a,*], Josef Bigun [b]

[a] Department of Computer Science, Addis Ababa University, Ethiopia
[b] School of Information Science, Computer and Electrical Engineering, Halmstad University, Sweden

ARTICLE INFO

Article history:
Received 29 September 2009
Available online 17 February 2011

Keywords:
Amharic
Ethiopic script
Handwriting recognition
Word recognition
OCR
HMM

ABSTRACT

This paper describes two approaches for Amharic word recognition in unconstrained handwritten text using HMMs. The first approach builds word models from concatenated features of constituent characters and in the second method HMMs of constituent characters are concatenated to form word model. In both cases, the features used for training and recognition are a set of primitive strokes and their spatial relationships. The recognition system does not require segmentation of characters but requires text line detection and extraction of structural features, which is done by making use of direction field tensor. The performance of the recognition system is tested by a dataset of unconstrained handwritten documents collected from various sources, and promising results are obtained.

1. Introduction

Amharic is the official language of Ethiopia which has a population of over 80 million at present. The language is believed to be derived from Geez, the liturgical language of Ethiopia since the 4th century AD. Amharic belongs to Afro-Asiatic language family, and today it has become the second most widely spoken Semitic language in the world, next to Arabic (Gordon, 2005). Along with other Ethiopian languages, Amharic uses Ethiopic script for writing. Ethiopic script has been in use since the 5th century BC (Gerard, 1981) and the recently standardized alphabet has a total of 435 characters, with several languages having their own special sets of characters representing the unique sounds of the respective languages. The Ethiopic script used by Amharic has 265 characters including 27 labialized characters (which are mostly representing two sounds, e.g. ሷ for ስዋ) and 34 base characters with six orders representing derived vocal sounds of the base character. The alphabet is written in a tabular format having seven columns where the first column represents the base characters and others represent their derived vocal sounds. The vowels of the alphabet are not encoded explicitly but appear as modifiers of the base characters, a characteristics of Semitic writing. Part of a handwritten alphabet is shown in Table 1.

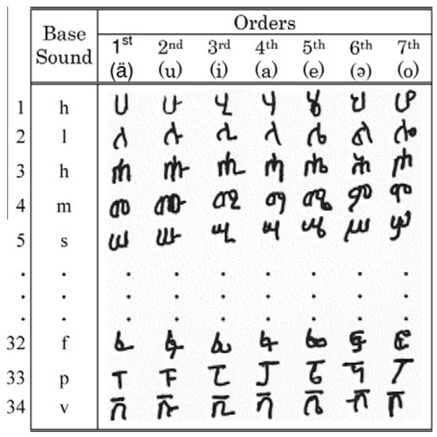
There are dozens of languages across the world with their own alphabets for writing. The advent of computing machines and the need for processing large volumes of data motivated research and development for automatic recognition of texts. Scripts with industrial and commercial importance received the earliest attention from researchers and developers of handwriting recognition. For example, offline recognition of Latin, Chinese, Japanese, Indian, and Arabic handwritten text has long been an area of active research and development (Arica and Yarman-Vural, 2001; Bunke, 2003; Lorigo and Govindaraju, 2006; Suen et al., 2003). However, Ethiopic handwriting recognition in general, and Amharic word recognition in particular, is one of the least investigated problems.

The purpose of automatic recognition of texts is to convert texts stored in a paper or other media to a standard encoding scheme representing the texts, e.g. ASCII or Unicode to the effect that efficient automatic services can be provided, e.g. searching in a text, postal distribution of letters, payment of checks, form extraction, etc. The conversion can be made online (at the time of writing) or offline (after writing is completed). Online recognition benefits from the temporal information captured when the text is written and better results are usually obtained as compared to its equivalent offline recognition. Offline text can be machine-printed or handwritten. Recognition of machine-printed text is considered to be a manageable problem. Several techniques applied on such documents are proved to be working in a wide range of real life applications for non-Ethiopic scripts (Mori et al., 1992; Srihari, 1992; Srihari et al., 1997) as well as Ethiopic script (Meshesha and Jawahar, 2005; Assabie and Bigun, 2007). However, offline recognition of unconstrained handwritten text is still one of the most challenging pattern recognition problems regardless of the writing system. The challenge mainly comes from cursiveness of handwriting, difficulty of detecting text lines, non-uniformity of spaces between characters and words, inconsistency of a writer, and variability in writing styles of different writers. In addition to the common problems pertinent to most scripts, the difficulty in recognition of Ethiopic handwriting also arises from the relatively

* Corresponding author.
E-mail addresses: yaregal@cs.aau.edu.et (Y. Assabie), josef.bigun@hh.se (J. Bigun).

0167-8655/$ - see front matter © 2011 Elsevier B.V. All rights reserved.
doi:10.1016/j.patrec.2011.02.007

Table 1
A sample of handwritten Ethiopic characters.



	Base Sound	Orders						
		1st (ä)	2nd (u)	3rd (i)	4th (a)	5th (e)	6th (ə)	7th (o)
1	h	ሀ	ሁ	ሂ	ሃ	ሄ	ህ	ሆ
2	l	ለ	ሉ	ሊ	ላ	ሌ	ል	ሎ
3	h	ሐ	ሑ	ሒ	ሓ	ሔ	ሕ	ሖ
4	m	መ	ሙ	ሚ	ማ	ሜ	ም	ሞ
5	s	ሠ	ሡ	ሢ	ሣ	ሤ	ሥ	ሦ
.	.	.	.	.	.	.	.	.
.	.	.	.	.	.	.	.	.
.	.	.	.	.	.	.	.	.
32	f	ፈ	ፉ	ፊ	ፋ	ፌ	ፍ	ፎ
33	p	ፐ	ፑ	ፒ	ፓ	ፔ	ፕ	ፖ
34	v	ቨ	ቩ	ቪ	ቫ	ቬ	ቭ	ቮ

large number of characters, their interclass similarity and structural complexity.

There are two paradigms in handwriting recognition: *segmentation-based* and *holistic* (Liu et al., 2003). Segmentation-based approach segments word images into constituent characters whereas holistic approach tries to recognize the whole word by ignoring character segmentation. Holistic approach also extracts representative features for the whole word, and it is more pragmatic in the case of cursive handwriting where characters are physically connected with each other and segmentation turns out to be impractical (Madhvanath and Govindaraju, 2001; Ruiz-Pinales et al., 2007). In both cases, recognition of unconstrained handwritten text remains a challenging task with the current technology. Consequently, several handwriting recognition techniques have been proposed over the years, with none of them providing high accuracy in unconstrained texts. Among the most commonly used methods are statistical approaches which include hidden Markov models (HMMs), Bayesian classifiers, support vector machines, fuzzy set reasoning, polynomial discriminate classifier, etc. (Arica and Yarman-Vural, 2001; El-Yacoubi et al., 1999; Jain et al., 2000; Liu and Fujisawa, 2008). For noisy data, neural networks showed good performances, and promising results have been reported for handwritten digit recognition (Cheriet et al., 2007; Marinai et al., 2005). Handwriting recognition is also achieved by using elastic matching which tolerates a certain range of geometric deformations of handwritten characters (Uchida and Sakoe, 2003). A structural approach is often applied for recognition by representing more complex structures using simpler graphical units and their relationships (Shi et al., 2003). Moreover, it has been shown that the use of multiple classifiers has a potential to improve recognition accuracy (Cheriet et al., 2007; Koerich et al., 2002). Recognition results can be further improved by the use of contextual information based on linguistic tools, e.g. analyzing at word level using spell checking techniques based on lexicon. Part of speech (POS) tagger also improves the recognition results by syntactically analyzing at sentence level (Fujisawa, 2008; Suen et al., 2003; Vinciarelli et al., 2004).

In this paper, we present Amharic word recognition in unconstrained handwritten text using HMMs. To the best of our knowledge, offline handwritten Amharic text has not been studied before. We also present a dataset of unconstrained handwritten Amharic text collected from various sources.[1] Currently, there are no publicly available datasets for such studies. The organization of the remaining sections of this paper is as follows. The basic theoretical backgrounds of HMMs and a review of their application to handwriting recognition is presented in Section 2. In Section 3, the proposed recognition approaches along with feature selection strategies are treated. Section 4 describes image processing, segmentation, and feature extraction techniques. Experimental results are reported in Section 5. We discuss about the overall recognition system and conclude in Section 6.

[1] The dataset is made available to the public and can be accessed by contacting authors.

2. Hidden Markov Models (HMMs)

Originally applied to the domain of speech recognition, HMMs have emerged as a powerful paradigm for modeling pattern sequences in different areas such as bio-informatics, gesture recognition, online handwriting recognition and online signature verification. Inspired by the success in such fields, they have also attracted a growing interest more recently in various computer vision applications including offline handwriting recognition (Plamondon and Srihari, 2000; Rabiner, 1989).

2.1. Problem statement

HMMs are doubly stochastic processes which model time varying dynamic patterns. The system being modeled is assumed to be a Markov process that is hidden (not observable), but can be observed through another stochastic process that produces the sequence of observations (Rabiner and Juang, 1986). The hidden process consists of a set of states connected to each other by transitions with probabilities, while the observed process consists of a set of outputs or observations, each of which may be emitted by states according to some output probability density function. Depending on the probability density functions, HMMs can be *discrete* or *continuous*. HMMs are characterized by the following parameters (Rabiner, 1989):

- N, the number of states in the model. Individual states are denoted as $S = \{S_1, S_2, \ldots, S_N\}$, where the state at time t is denoted by the variable q_t and it takes one of the states in the set S as value.
- M, the number of distinct observation symbols per state, denoted as $V = \{v_1, v_2, \ldots, v_M\}$.
- $A = \{a_{ij}\}$, the state transition probability distribution where $a_{ij} = P(q_{t+1} = S_j | q_t = S_i)$, $1 < i, j \leqslant N$.
- $\boldsymbol{B} = \{b_j(k)\}$, the observation symbol probability distribution in state j, where $b_j(k) = P(v_k \text{ at } t | q_t = S_j)$, $1 \leqslant j \leqslant N$, $1 \leqslant k \leqslant M$.
- $\pi = \{\pi_i\}$, the initial state distribution, where $\pi_i = P(q_1 = S_i)$, $1 \leqslant i \leqslant N$.

The above HMM is represented by a compact notation:

$$\lambda = \{A, B, \pi\} \tag{1}$$

For the HMM model with the compact notation, there are three basic problems that must be solved: *evaluation*, *decoding*, and *training* problems. Fortunately, the theories behind HMMs are based on strong statistical and mathematical foundations which solve the stated problems. The evaluation problem is solved by *forward–backward procedure*; the decoding problem can be solved by using the *Viterbi algorithm*; and the training problem is solved by *Baum-Welch algorithm*. Further details are presented in (El-Yacoubi et al., 1999; Rabiner, 1989; Young et al., 2006).

2.2. Application to handwriting recognition

There is a growing number of researches investigating the application of HMMs for handwriting recognition. The reward comes from the fact that HMMs have strong theoretical and statistical foundation to cope with noise and variability of data, of which handwritten text is a typical example. Recognition of isolated handwritten characters can be done using HMMs by building a model for each character. Since the number of characters in a script is not very large, it is possible to collect sufficient training data for each class. The same is true for a small set of words in a specific application such as recognition of city names and bank checks, where sufficient training data can be made available and a model can be built for each word. In the case of general purpose handwritten text recognition, the number of words becomes huge resulting in a problem to collect sufficient training data and build HMM for each word. Thus, the traditional way of using HMMs for handwritten word recognition is by concatenation of HMMs of characters constituting the word (El-Yacoubi et al., 1999; Koerich et al., 2003). Like other recognition methods, HMM-based recognition systems usually require preprocessing procedures such as slant correction and size normalization. After normalization, input features are usually extracted by moving a sliding window in the image from left to right to produce a sequence of observations. The features are extracted in each window frame using image transformations, such as *cosine transform*, *Fourier transform*, and *Karhunen–Loève transform*.

3. The proposed recognition system

In our method, we use structural features of characters as the building blocks of the recognition system. We propose two methods of recognition strategies both of which are using these structural features. In both cases, recognition of a word image is made by decoding the hidden states for the observed sequences of word-level structural features. The components of the recognition system are discussed below in detail.

3.1. Feature design

The design of suitable features is one of the most important factors in achieving good recognition results. It should be made in such a way that features represent the most relevant information for the classification purpose at hand. Here is it determined by minimizing the intra-class pattern variability while enhancing the inter-class pattern variability. In this work, feature vectors are computed from the structural features, i.e. primitive strokes and their spatial relationships, which are extracted in sequential order based on their spatial arrangements. Primitive strokes are formed from vertical and diagonal lines and end points of horizontal lines, whereas connectors are defined as horizontal lines between two primitives. Primitive strokes for handwritten characters are hierarchically classified based on their orientation/structure type, relative length within the character, and relative spatial position. A description of similar features is exposed in (Assabie and Bigun, 2007) where they were first used for multifont and size-resilient recognition of machine-printed Ethiopic characters. For the purpose of computation, each classification level is assigned numbers as labels ranging from 6 to 9. The hierarchy of classification is given as follows.

i. *Orientation/structure type*: There are three groups of orientations for primitive strokes namely, *forward slash* (label[2] 9), *vertical* (8), and *backslash* (7). *Appendages* (6) do not fit to a specific orientation. Rather, they are recognized by their structure type in the case of machine printed text, e.g. in where there are three appendages placed at the end of horizontal lines. However, in handwritten text, appendages are usually not marked well and we de ne them as the end points of horizontal lines as in .
ii. *Relative length*: The orientation of primitives is further classified based on their relative length as *long* (9), *medium* (8), and *short* (7). Long is defined as a primitive that runs from the top to the bottom of the character, where as short is a primitive that touches neither the top nor the bottom of the character. Medium refers to a primitive that touches either the top or the bottom (but not both) of the character. Due to their small size, appendages are always considered as short.
iii. *Relative spatial position*: At this level of classification hierarchy, primitives are further classified according to their spatial position with in the character as *top* (9), *top-to-bottom* (8), *bottom* (7), and *middle* (6). Short primitives can only have a relative spatial position of middle. Top-to-bottom position applies to long primitives which run from the top to the bottom of the character. Primitives with medium relative size can have either top or bottom spatial position. Appendages may appear at the top, middle, or bottom of the character.

The above classification scheme results in 15 types of primitive strokes, which are used to represent all the 435 Ethiopic characters. Table 2 summarizes lists of these primitive strokes and their numerical codes. The example characters in the table contain several primitive strokes, but only one of them pointed by arrows illustrates the respective entry. The example primitive strokes for each entry are further indicated in the last column where they are extracted from the respective example characters. Note that horizontal lines are classified as connectors between two primitive strokes, and only their endpoints are classified as appendages which are shown as *dots* in the primitive strokes column.

As aforementioned, there exist horizontal strokes but these are evidences of connections between two primitives. The way two primitives are connected to each other with horizontal lines is referred to as *spatial relationship*. A primitive can be connected to another at one or more of the following regions: *top* (1), *middle* (2), and *bottom* (3). A connection between two primitives is represented by *xy* where *x* and *y* are numbers representing connection regions for the left and right primitives, respectively. Between two primitives, there can also be two or three connections, and a total of 18 spatial relationships are identified as shown in Table 3. The first connection found as one goes from top to bottom of connected primitives is defined as *principal connection*. There are a total of nine principal connections where only three of them (11, 12 and 21) allow additional connections which are termed as *supplementary connections*.

A spatial relationship between two primitives is defined to have six feature values where a value of zero is padded at the beginning for those whose number of connections are two or less. For example, the feature value of a spatial relationship of the type 13 () will be $\{0,0,0,0,1,3\}$. The sequential order of primitive strokes *A* and *B* is represented as *AB* if *A* is spatially located at the left or top of *B*. Each primitive is connected to another one to the left except the first primitive in each character, in which case it does not have any one to be connected to the left. In such cases, all the six feature values for such spatial relationship will be all zeros.

3.2. The character feature list

The character feature list stores possibly occurring sequences of primitive strokes and their spatial relationships for each character in the alphabet. Each primitive stroke appearing in a character is represented by a feature vector of nine digits of which the first six are for the spatial relationships and the last three are for the primitive strokes. Thus, a single sample of a character is repre-

[2] For convenience we drop the word *label* in the sequel when it is clear from the context.

Table 2
Classification of primitive strokes.

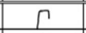
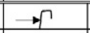
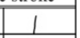
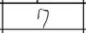
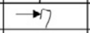
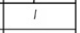
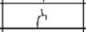
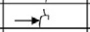
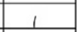
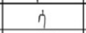
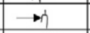
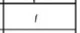
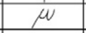
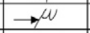
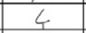
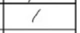
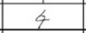
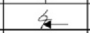
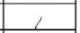
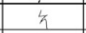
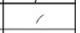
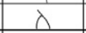
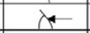
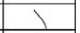
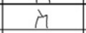
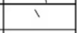
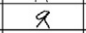
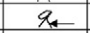
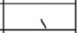
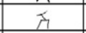
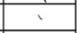
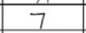
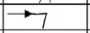
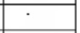
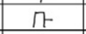
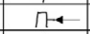
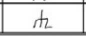
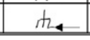
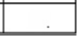
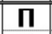
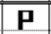
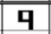
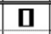
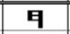
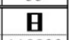
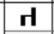
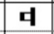
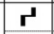
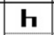
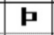
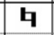
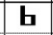
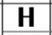
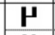
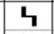
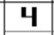
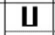
Orientation/ Structure	Length	Position	Numerical code	Example		
				Character	Primitive stroke	
Vertical (8)	Long (9)	Top-to-bottom (8)	8 9 8			
	Medium (8)	Top (9)	8 8 9			
		Bottom (7)	8 8 7			
	Short (7)	Middle (6)	8 7 6			
Forward Slash (9)	Long (9)	Top-to-bottom (8)	9 9 8			
	Medium (8)	Top (9)	9 8 9			
		Bottom (7)	9 8 7			
	Short (7)	Middle (6)	9 7 6			
Backslash (7)	Long (9)	Top-to-bottom (8)	7 9 8			
	Medium (8)	Top (9)	7 8 9			
		Bottom (7)	7 8 7			
	Short (7)	Middle (6)	7 7 6			
Appendage (6)	Short (7)	Top (9)	6 7 9			
		Middle (6)	6 7 6			
		Bottom (7)	6 7 7			

Table 3
Connection types between primitives.

Principal connection	Number of supplementary connections					
	None	One			Two	
					22	22
		23	32	33	32	33
11	 11	 1123	 1132	 1133	 112232	 112233
12	 12		 1232			
13	 13					
21	 21	 2123	 2132	 2133		
22	 22					
23	 23					
31	 31					
32	 32					
33	 33					

sented by sequences of feature vectors where each vector has nine-digit values. The character feature list is a collection of such sequences of feature vectors generated by various sample characters. A character can have many sample features stored in the character feature list reflecting variations of writing styles and slants. This helps to train the system with slanted characters as well, and as a result it does not require slant correction later in the image preprocessing stage. Fig. 1 illustrates different handwritten symbols for the character.

3.3. Training and recognition

The goal of the training phase is to estimate the parameter values of models from a set of training samples, and the recognition phase decodes the input data based on the observation sequence. Below we discuss the two proposed methods: feature-level and HMM-level concatenation. In both cases, Baum–Welch algorithm is used for training and Viterbi algorithm is used for recognition.

3.3.1. Feature-level concatenation method

In this method, training samples for a given word are generated from a character feature list which stores possibly occurring sample features for each character. Suppose that the input word W has sequences of characters $C_1, C_2, C_3, \ldots, C_m$, where m is the total number of characters making up the word. Then, sample features of the word are generated as all combinations of sample features of each character, yielding w sample features of the word computed as:

$$w = \prod_{i=1}^{m} n(C_i) \qquad (2)$$

where $n(C_i)$ is the total number of samples for character C_i. Fig. 2 shows a sample feature for the word እገር generated from the character feature list. Each group in the rectangular box beneath characters represents sample features for the corresponding character, whereas each line represents a feature vector of primitives and their associated spatial relationships.

After generating sample features for the input word, the next procedure is HMM initialization which sets a prototype for HMM of the word to be trained including its model topology, transition and output distribution parameters. A simplified flowchart of training and recognition procedures is shown in Fig. 3. The dotted-line box in the flowchart shows repetitive tasks for each word.

Gaussian probability function that consists of means and variances is used to define the model parameters. The number of states of a word corresponds to the total number of primitive strokes in the word. The HMM topology of እገር which has eight primitive strokes is shown in Fig. 4. Once the HMM is trained with sample features of the word, the model is stored into a master model file which will be used later during the recognition phase.

In the recognition phase, handwritten text image is processed to detect lines and segment words. For each word in the text image, a sequence of primitive strokes and their spatial relationship is extracted. Fig. 5 shows primitive strokes and spatial relationships identified for the handwritten word እገር Then, the sequence is generated as: $\{\{\alpha A, \beta B, \gamma \Gamma\}, \{\delta \Delta, \epsilon E\}, \{\zeta Z, \eta H, \mu M\}\}$, where the Greek capital letters represent primitive strokes and smaller letters rep-

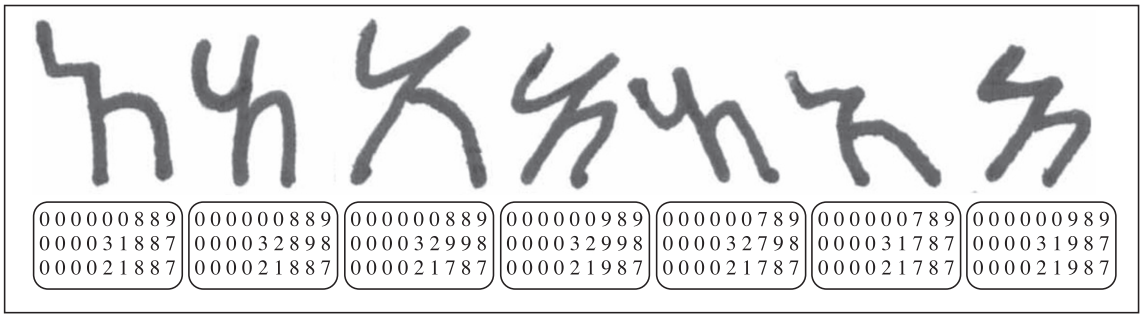


Fig. 1. Various handwritten symbols for the character አ.

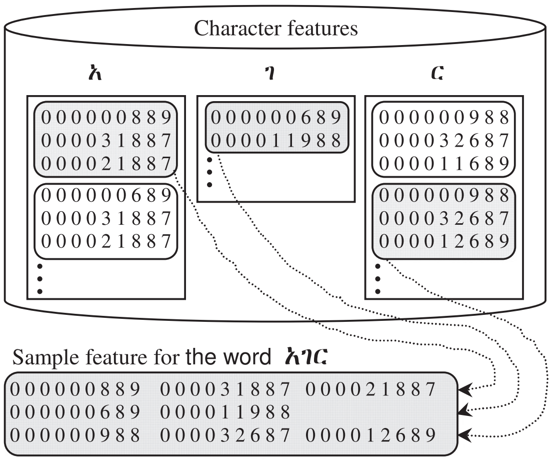


Fig. 2. Generation of sample feature for the word አገር.

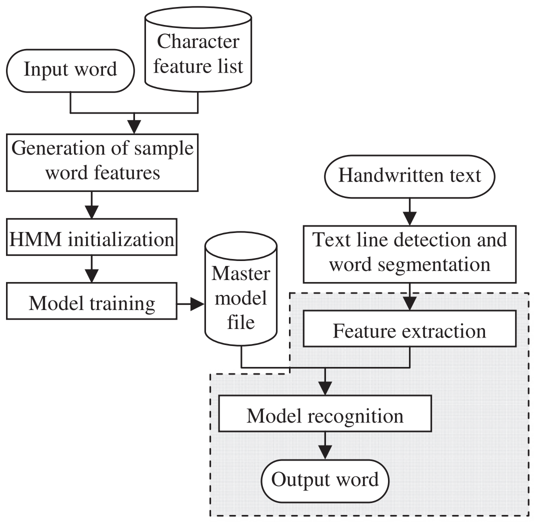


Fig. 3. Training and recognition flowchart in feature-level concatenation method.

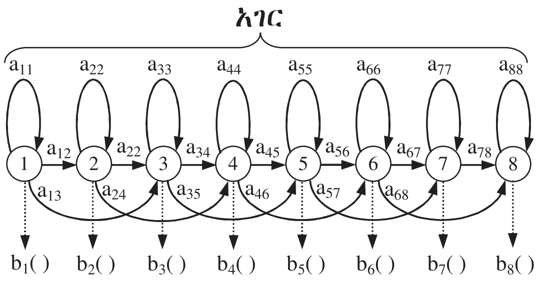


Fig. 4. HMM topology for the word አገር in feature-level concatenation.

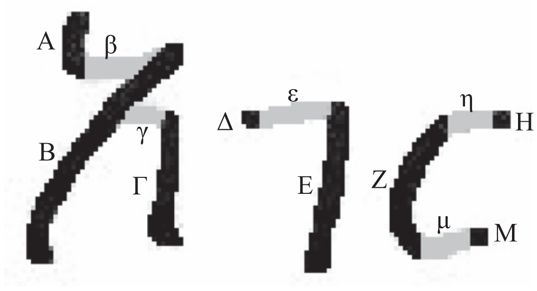
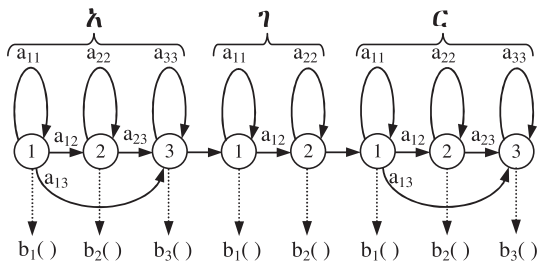


Fig. 5. Structural features for the handwritten word አገር.

resent associated spatial relationships. Note that α, δ, and ζ are not shown in the figure since they correspond to a spatial relationship of the first primitive stroke in each character, in which they do not have primitive to the left to be connected with. Once structural features are identified and classified, they are assigned with features values as discussed in Section 3.1. Then, the extracted feature sequences are considered as observations which are used by the decoder for recognition. For the most part of Amharic words, a single model is built for each of them. However, comparatively few words can also have more than one models for two reasons. The first is because of characters (along with their derivatives) which represent the same sound but different shapes. Groups of base characters representing the same sound are: {ሀ, ሐ, ኀ, ኸ}, {ሠ, ሰ}, and {ጸ, ፀ}. A word containing the sounds of such characters and/or their derivatives can be written in several ways. For example, one can write the word ታህሳስ also in various ways as: ታሕሳስ, ታኅሳስ, ታኸሳስ, ታህሳሥ, ታኅሣስ, ታህሣስ, ታህሣሥ, ታሕሣሥ, ታኸሣሥ, etc. While some of them may look awkward to native users, they are not considered as wrong, and readers can still get the same meaning. However, as far as the recognition system is concerned, they have different features and consequently they are treated as different words. Therefore, despite the same sound and meaning, different models are built for each of them. The second reason is that there are some variations in writing styles of a character resulting in different number of primitives to be extracted for the same character. For example, handwritten symbols ሥ, ሥ, and ሥ represent the same character ሥ yielding 2, 4, and 5 primitive strokes, respectively. Accordingly, the number of states of HMMs for words containing this character will change. Thus, for such cases where a single word with the same set of characters results in two or more models, the word is temporarily given different word codes with



አ
ገ
ር
a11 a22 a33 a11 a22 a11 a22 a33
a12 a23 a12 a12 a23
a13 a13
1 2 3 1 2 1 2 3
b1( ) b2( ) b3( ) b1( ) b2( ) b1( ) b2( ) b3( )

Fig. 6. HMM topology of አገር using HMMs of constituent characters.

respect the models. They are treated as different words in the training and decoding phases, but will be given the same label in the output. Apart from building multiple models, such words do not incur extra complexities on training and recognition.

3.3.2. HMM-level concatenation method

While the training and recognition procedures remain similar to that of feature-level concatenation method, the basic idea in HMM-level concatenation method is that HMMs are built for each character from the stored sample features of characters, and the word model is made up of the concatenation of the constituent character HMMs. Given a sequence of characters for a word, the final state of a character HMM is connected to the initial state of the next character. The number of primitives in a character corresponds to the states of its HMM. Two or more models could be built for a single character based on the number of primitives that character is formed from. Taking the above examples ሥ, ሥ, and ሥ which represent the same character ሥ, the character has three HMMs having 3, 4, and 5 states with respect to the handwritten symbol types. Fig. 6 shows the HMM topology for the word አገር formed by concatenation of the HMMs of constituent characters.

3.4. Adaptability

The proposed recognition system can be extended to accommodate new words. Since word features are generated from the feature lists of constituent characters, the system does not require actual examples of handwritten words. For any given word, the initial parameters are automatically set by the system and does not require manual intervention, e.g. the number of states are computed from the total number of primitives the word comprises. Therefore, it is possible to train words easily for a specific set of new applications like postal address recognition and bank check processing. Similarly, the same procedure can be taken to use the system for other Ethiopian languages. The requirement here is to make sure that structural features for special character sets corresponding to the respective languages are added in the character feature list.

4. Image processing and feature extraction

Feature extraction is an essential part of pattern recognition systems with a direct effect on the result. In offline handwriting recognition, feature extraction involves image analysis and processing. One of the most commonly used filters for image processing are Gaussian filters and derivatives of Gaussians. Gaussian filter is frequently used as a low-pass filter for noise suppression and Gaussian derivatives are used to detect and localize edges along with determining their orientation (Basu, 1994). The use of Gaussian kernels for image processing has become popular, among others, due to (i) their separability in the x and y directions, and (ii) directional isotropy, i.e. circular symmetry (Bigun et al., 2004). Mathematically, a 2D Gaussian kernel is defined as:

$$g(x,y) = \frac{1}{2\pi\sigma^2} \exp\left(-\frac{x^2+y^2}{2\sigma^2}\right) \tag{3}$$

where σ is the standard deviation. With its separability property, the 2D Gaussian is more efficiently computed as convolution of two 1D Gaussians, $g(x)$ and $g(y)$, which are defined as follows

$$g(x) = \frac{1}{\sqrt{2\pi}\sigma^2} \exp\left(-\frac{x^2}{2\sigma^2}\right) \tag{4}$$

$$g(y) = \frac{1}{\sqrt{2\pi}\sigma^2} \exp\left(-\frac{y^2}{2\sigma^2}\right) \tag{5}$$

with the abuse of notation that g is used to mean both a 2D function $g(x,y)$ when it has two arguments, and a 1D function $g(\tau)$, when it has one argument. In this work, we use gradient and direction fields as tools for separating texts from background, detecting text lines and extracting features. We use Gaussians and derivatives of Gaussians as filtering tools in the image processing phase. Below, we give a brief summary of both the gradient field and the direction field tensor. The later is introduced by Bigun and Granlund (1987) and exposed in further detail in (Bigun, 2006).

4.1. Gradient field

Gradient field has been used as a traditional tool over many years for feature extraction in image analysis problems (Kim et al., 1998; Tan et al., 1996). Gradient is a vector representing the change in gray level in two orthogonal directions. This can be calculated by taking the difference in value of neighboring pixels in a given pair of orthogonal directions, producing a vector for each pixel. The magnitude of the vector at each pixel measures the amount of change in intensity, and the angle of the vector shows the direction of maximal intensity changes and can be expressed in the range of $[0\ldots360)$ degrees. For a local neighborhood $f(x,y)$ of an image f, the gradient field ∇f is computed by using Gaussian derivative operators D_x and D_y

$$\nabla f(x,y) = (D_x + iD_y)f(x,y) = \sum_j f_j (D_x + iD_y) g(x - x_j, y - y_j) \tag{6}$$

and sampling the image at (x_j, y_j). It amounts to a convolution with a derivative of Gaussians. The complex partial derivative operator $D_x + iD_y$ is defined as:

$$D_x + iD_y = \frac{\partial}{\partial x} + i\frac{\partial}{\partial y} \tag{7}$$

and used instead of the vector representation, as it has some notational advantages that will explained further below.

4.2. Direction fields

A local neighborhood with ideal local direction is characterized by the fact that the gray value remains constant in one direction (along the direction of lines), and only changes in the orthogonal direction. Since the directional features are observed along lines, the local direction is also called *Linear Symmetry (LS)*. The LS property of an image can be estimated by analyzing the direction field tensor (Bigun, 2006; Bigun et al., 2004). The direction tensor, also called the structure tensor, is a real valued triplet, which is a tensor representing the local directions of pixels. For a local neighborhood of an image $f(x,y)$, the direction tensor, also called the structure tensor S, is computed as a 2×2 symmetric matrix using Gaussian derivative operators D_x and D_y.

$$S = \begin{pmatrix} \int\int (D_x f)^2 dx\,dy & \int\int (D_x f)(D_y f) dx\,dy \\ \int\int (D_x f)(D_y f) dx\,dy & \int\int (D_x f)^2 dx\,dy \end{pmatrix} \tag{8}$$

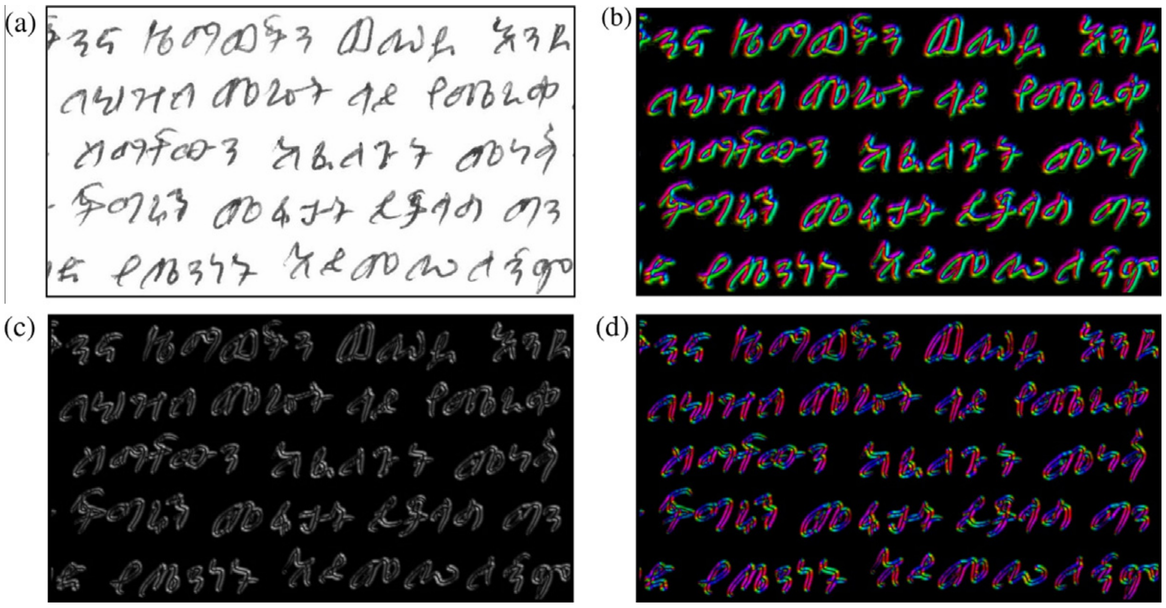


Fig. 7. (a) Handwritten Amharic text, (b) ∇f, (c) I_{11}, and (d) I_{20} of a.

Linear symmetry exists among others at edges where there are gray level changes and an evidence for its existence can be estimated by eigenvalue analysis of the direction tensor or equivalently by using complex moments of order two which are defined as follows:

$$I_{mn} = \int\int ((D_x + iD_y)f)^m ((D_x - iD_y)f)^n dx\,dy \tag{9}$$

where m and n are non-negative integers. Among other orders, of interest to us are I_{11}, and I_{20} derived as:

$$I_{11} = \int\int |(D_x + iD_y)f|^2 dx\,dy \tag{10}$$

$$I_{20} = \int\int ((D_x + iD_y)f)^2 dx\,dy \tag{11}$$

I_{11} is a scalar value that measures the amount of gray value changes in a local neighborhood of pixels and equals to the sum of eigenvalues of S. The value of I_{20} is a complex number where the argument is the local direction of pixels in double angle representation (the direction of major eigenvector S) and the magnitude is a measure of the local LS strength (the difference of eigenvalues of S). As illustrated in Fig. 7b and d, gradient and direction field images can be displayed in color where the hue represents direction of pixels (in double angle representation in the case of direction field tensor image) with the red[3] color corresponding to the direction of zero degree. In Fig. 7d, the region with black color represents pixels with low magnitudes and thus they are said to be lacking LS properties. Whether a given pixel lacks LS property is determined by a threshold value. The scalar value I_{11} can also be displayed as gray scale image as shown in Fig. 7c.

In this work, we use ∇f and I_{20} for image processing and analysis. The motivation of using both directional information comes from the respective reward they provide. The advantage of ∇f over I_{20} is that its argument is computed with directions of pixels expressed with $[0\ldots360)$ degrees, representing the left and right edges (black-white, white-black transitions) differently. In the case of I_{20}, the resultant argument corresponds to the range of $[0\ldots180)$ degrees, with opposite edges mentioned not being discerned. While the direction information of I_{20} is "phase-free", smoothing of it in a window does not lead to cancelation effects from which ∇f suffers. Another advantage of I_{20} is that it automatically encodes the optimal direction in the total least square error sense as this optional fitting is implicitly carried out by (tensor) averaging. The averaging process amplifies linear structures and suppresses non-linear structures automatically. The gradient field, i.e. ∇f simply computes the differences in the intensity of pixels, without attempting to fit an optimal direction in the total least square error sense. Their difference is clearly visible when using them in noisy images. We use the synergy of both ∇f and I_{20} for the following image processing procedures.

4.3. Text line detection and word segmentation

Text line detection and word segmentation are among the most critical sub-processes in character recognition. In the case of handwritten documents, there is no uniformity: text lines may not be straight, the gap between words may vary greatly, and characters may be physically connected. Because of these problems, automatic segmentation poses a challenge for researchers in handwriting recognition. Thus, studies are still being carried out to detect text lines (Li et al., 2006), and segment words (Huang and Srihari, 2008) and characters (Selvi and Indira, 2005) in handwritten documents. Although there are also other state of the art research works on text line detection and word segmentation (Bar-Yosef et al., 2009; Louloudis et al., 2009), we propose a new text line detection and word segmentation technique that uses the direction field image. The advantage here is that we continue to be working on the resultant image that is computed to be used for feature extraction as well. Although, our proposed recognition system does not require the segmentation of characters, it is generated as a byproduct in the process of text line extraction and word segmentation. In the process, physically connected characters are emerging as a single character, whereas parts of a character which are not connected with each other are over-segmented as several parts. In either of the cases, a single segmented unit is termed as *pseudo-character*, but hereafter we simply refer to them as characters. The final result of this process is that the background is separated from the foreground (text). The proposed algorithm which extracts text lines and segment words from the direction field

[3] For interpretation of color in Fig. 7, the reader is referred to the web version of this article.

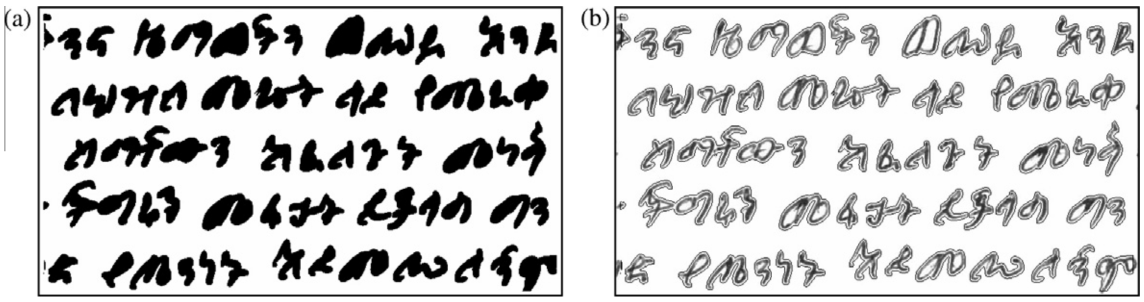


Fig. 8. (a) Character regions separated from the background, (b) character segmentation mapped onto the original text.

(I_{20}) image passes through two phases. In the first pass, the image is traversed from top to down and pixels are grouped into two as *blocked* and *open* regions. In the process, a pixel is sequentially classified as open if it:

- is in the first row of the direction field image,
- lacks LS property and its immediate top pixel is open,
- lacks LS property and one of its sideways neighborhood is open.

The remaining are grouped as blocked pixels. As the scanning progresses each group of interconnected blocked pixels are designated as character regions, and open pixels form the background. Fig. 8a shows blocked and open pixels for the Amharic handwritten text of Fig. 7a, with the black color representing blocked pixels and white representing open pixels. As a result of such pixel classification, ultimately we get foreground (character regions) separated from the background. The boundaries of blocked pixels form borders of segmented characters. Fig. 8b shows such character segmentation results for the text shown in Fig. 7a.

In the second pass, the resultant image (with the character region separated from the background) is traversed from left to right grouping each segmented character into appropriate text lines. As the traversal proceeds, the global and local directions of each text line are computed. The *global* direction is the average direction of the line passing through all member characters, and the *local* direction is computed from few characters at the head of text lines. The global and local directions help predict the direction in which the next member character is found. This is essential especially in skewed documents. During traversal, when a segmented character is found, existing text lines compete for the character, and a text line is selected based on its proximity and legibility to the character. If the character is not within the local and global directions of the candidate text line, then the character forms a new line. This text line detection technique tolerates skewed documents because we follow the directions of text lines. Fig. 9 shows an automatic line detection in a skewed handwritten document. Words are then segmented based on the relative gap R between characters within text lines, defined as:

$$R_i = G_i - G_{i-1} \tag{12}$$

where G_i is the horizontal gap between the ith character and its predecessor. Although the horizontal gap between consecutive characters varies greatly, the relative gap suppresses those variations and a fixed threshold fairly segments words. It should also be noted that, as this procedure does not use contextual information, word segmentation will fail if gaps between characters in a word are not slightly greater than gaps between words.

4.4. Feature extraction

Features of segmented words are extracted based on the optimal direction of pixels. Once character regions are separated from the background, as discussed in the previous section, the boundaries of characters also form the boundaries of their primitive strokes and connectors except in the case of holes. Holes are formed as a result of two or more connections between primitive strokes in a character. As shown in Fig. 10b, which is a result of the character segmentation process applied on Fig. 10a, there are four holes which are not yet identified as primitive strokes. Further processes to identify constituent primitive strokes forming holes are done using the combination of ∇f and I_{20} images. The magnitude of the I_{20} and the angle of ∇f are combined resulting in optimized linear structures expressed with the range of [0...360) degrees. Noting that strokes in characters produce two edges (left and right edges for vertical strokes, and top and bottom edges for horizontal strokes), the angle of ∇f discriminates the two edge types. The magnitude of I_{20} is used to classify whether a local neighborhood forms a linear structure or not. By mapping the character boundaries on the combined directional image, as shown in Fig. 10c, strokes are identified for each character of a given word image. Fig. 10d illustrates extracted strokes from a word. Strokes are further classified as primitive strokes and connectors based on the direction information of their pixels. In the ∇f angle, stroke pixels with directions (60...120) or (240...300) degrees are set as parts of connectors and the rest are considered as parts of primitive strokes. Primitives are then further classified and assigned with feature values using their direction, relative length, and spatial

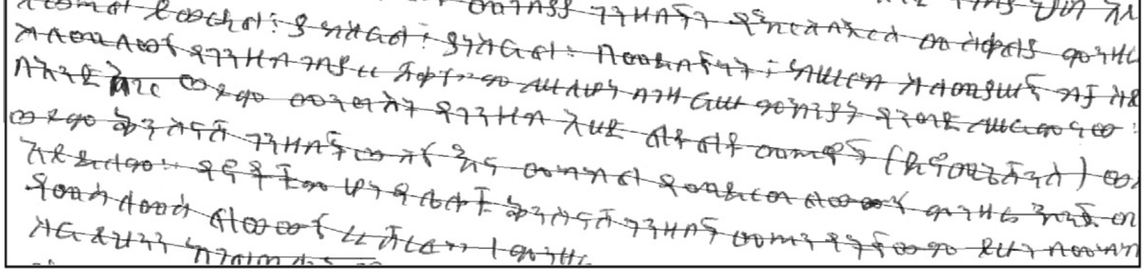


Fig. 9. Text line detection.

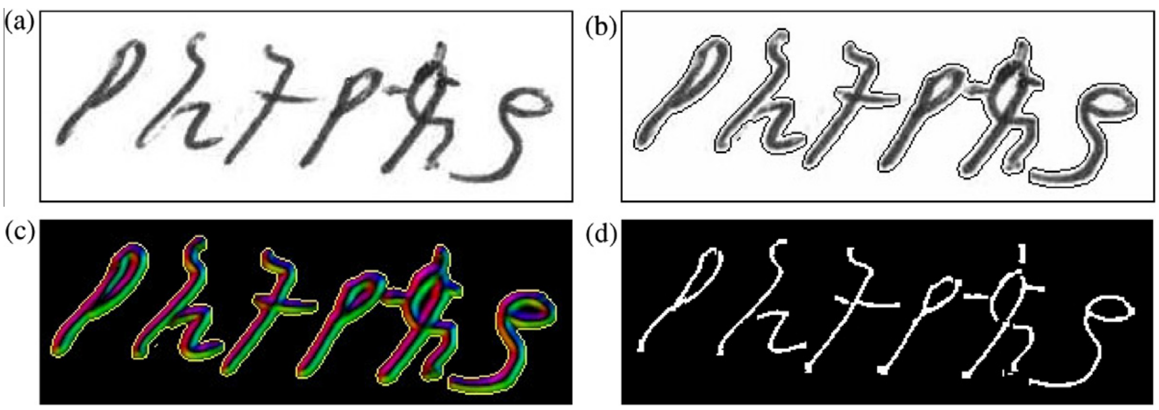


Fig. 10. Feature extraction process; (a) handwritten text, (b) segmentation result, (c) segmentation superimposed on the synergy of ∇f and I_{20} images, and (d) extracted features.

position. Further details on the procedures of feature extraction are exposed in (Assabie and Bigun, 2007).

5. Experiments

Recognition result of handwritten documents in general highly depends on the characteristics of documents with respect to image quality and readability. Such variations make comparison of handwritten recognition systems more difficult even for a single script. To curb this problem, datasets of handwritten documents for various scripts, which are used as a benchmark for comparison of recognition performances, have been developed. To the knowledge of authors, however, there is no dataset of handwritten Ethiopic documents for any of Ethiopian languages including Amharic. Therefore, we collected a dataset of handwritten Amharic documents, which is described more in detail below. The size of Gaussians used for filtering operations in the image processing phase depends on the characteristics of images. For noisy documents a symmetric Gaussian window of 5×5 pixels (standard deviation of 0.83) was used. The same size was used for texts with bigger character sizes. However, for texts with small character sizes 3×3 window (standard deviation of 0.50) was used to avoid over-smoothing. Training and recognition of HMMs were implemented by using the HTK toolkit (Young et al., 2006).

5.1. Dataset collection

The dataset of handwritten Amharic documents we developed is collected from 177 writers. The writers were provided with Amharic documents dealing with various issues and they used ordinary pen of their own and white papers they are provided for writing. The dataset is meant to reflect a real world set of documents. Writers were oriented to write freely without any constraint as they used to. However, some of them made their writing even more compact than the usual as they tried to complete a given text on a limited number of papers they are provided. A total of 307 pages were collected and scanned at a resolution of 300 dpi, from which we extracted 10,932 distinct words to build a list of words for training. The dataset is divided into approximately two equal parts as poor and good quality images based on their readability, clarity and strength of pen inks, cursiveness and noises. Along with such parameters, the proportion of the classific In addition, the dataset consists of isolated characters where another group of 152 writers have participated. Each participant writes all the 265 Amharic character set three times resulting in a total of about 120,840 isolated character samples included in the dataset. While the primary goal is to set a standard for testing isolated handwritten character recognition systems, we also used the samples to extract features of characters which in turn is used to form word features by concatenating features of constituent characters. In the case of HMM-level concatenation method, character HMMs are trained based on the features extracted from isolated characters. Although they have not been used for training and testing, 114 pages of Geez and Amharic handwritten texts from the Ethiopian Orthodox Church are also included in the dataset. Samples of images from the dataset are shown in Fig. 11.

5.2. Results

Recognition rates show expected variations due to the quality of the handwriting and the size of training words. To present a fair evaluation of the system, we tested it using documents classified as good and bad quality images in the dataset. In addition to the whole word list extracted from the dataset, the most frequent 10 and 100 words were also used for training and testing the system. The results are summarized in Tables 4 and 5.

5.3. Discussion

HMM concatenation method has been used widely in HMM-based handwritten word recognition systems. The results reveal that feature-level concatenation method consistently performs

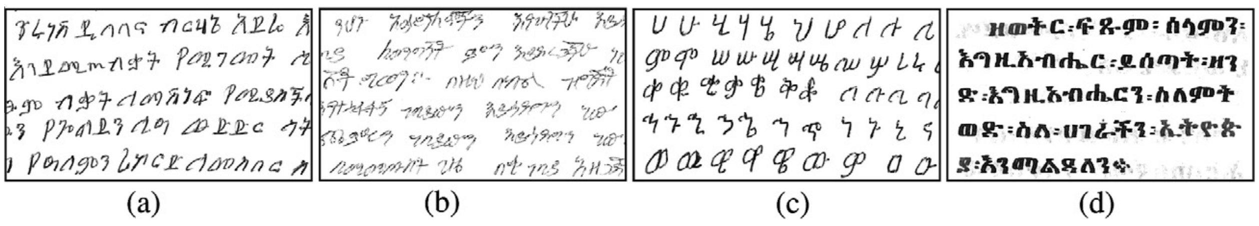


Fig. 11. Sample images from the dataset; (a) good quality text, (b) poor quality text, (c) isolated characters, and (d) church document.

Table 4
Recognition result for feature-level concatenation method.

Quality of text	Number of training words		
	10	100	10,932
Good	98%	93%	76%
Poor	85%	81%	53%

Table 5
Recognition result for HMM-level concatenation method.

Quality of text	Number of training words		
	10	100	10,932
Good	92%	86%	66%
Poor	78%	73%	41%

better than HMM-level concatenation across different document qualities and varying sizes of training and test data. A major factor discouraging feature-level concatenation method is the unsuitability of collecting sufficient training data, especially in the case of large lexicon sizes. In our approach, this problem is overcome by automatically generating sufficient training word sample features from the feature sets of constituent characters. The major drawback of the feature-level method comes during the training phase which takes more time as compared to the discussed alternative. In fact, only a smaller number of HMMs amounting to the total number of characters need to be trained in the case of HMM-level concatenation as opposed to feature-level concatenation where the HMMs are equal to the number of words. The disparity grows as the size of training and test words grow. Both methods are tolerant of the connection between characters in a word. Two connected characters will not disrupt the number of states of the word, and it is merely considered as a slight change in the feature values. The effect is reflected only on the connection type of the first primitive stroke in the second character. However, extra strokes added to characters in a word hampers its recognition as it induces an extra state for each extra stroke. Extra strokes arise when writers are writing in various gothic styles. Whereas it is possible to include the most common types of such character styles in the stored feature list, it is still difficult to include them exhaustively.

As shown in the recognition results, good quality documents show better results than poor quality documents. The difference in recognition results arises from various levels of processes such as word segmentation and feature extraction. Text line detection works well for both classes of documents (more than 97% accuracy is achieved even in skewed documents). Word segmentation in poor quality documents fails when the space between words is not sufficient enough for segmentation, e.g. due to ligatures and noises. Feature extraction becomes a source of error in recognition of poor quality documents with strong noise and/or low intensity of pen inks. For noisy documents, non-removeable noise could be considered as a stroke which ultimately yields wrong number of states during the HMM modeling. In the case of low intensity of pen inks, some part of the strokes could get over-smoothed causing a single stroke to be broken into two or more. It means that additional states are induced for each broken stroke which ultimately hinders recognition of the word. Although ligatures in a word will not change the number of states of the word, they slightly change the feature values of states. Thus, recognition could also fail if there are many ligatures in a word.

5.4. Comparison

Although it may not be directly relevant to compare handwritten recognition systems of various scripts which are performed under different experimental conditions, we present three other works whose methods are related to this paper as an indication of progress for Ethiopic handwriting case. An Amharic handwriting word recognition system was presented by Assabie and Bigun (2008). The system uses a lexicon to optimize character level recognition results and top-1 choice of 73% accuracy was achieved for good quality documents and the lexicon size does not have impact on the recognition rate. Thus, the HMM-based approaches become preferable for special applications which depend on smaller sizes of lexicons such as bank check processing and address recognition. Recognition of handwritten Arabic words using HMM was described by Khorsheed (2003), and reported recognition rates with top-1 choices of 72% without spell-checker and 87% with spell-checker. A lexicon-driven HMM was also used by Koerich et al. (2003) for Latin script with bigram probabilities and reported for various lexicon sizes. The top-1 choice recognition results for lexicon sizes 10, 100, 1 000, 10 000, and 30, 000 were approximately 99%, 95%, 89%, 78%, and 71%, respectively.

6. Conclusion

We presented feature-level and HMM-level concatenation of characters for recognition of handwritten Amharic words. The recognition system also includes script-independent text line detection, and character and word segmentation processes. For feature-level concatenation method, sample features of training words are generated by feature sets of constituent characters. The feature set stores a variety of sample features for each character reflecting different real-world writing styles. The advantage of this is that it is possible to produce real-world sample word features for any word without collecting sample text, which has turned out to be writer-independent recognition system. It also means that the system can be directly applied for other Ethiopian languages which use Ethiopic script for writing. Since we are encoding the relative size of primitive strokes, recognition system does not require size normalization. The recognition result can be further improved by employing language models in HMMs. The database we developed can be used as a benchmark resource for further studies on recognition of Ethiopic script.

Acknowledgement

The financial support of SIDA (Swedish International Development Agency) is gratefully acknowledged.

References

Arica, N., Yarman-Vural, F.T., 2001. An overview of character recognition focused on off-line handwriting. IEEE Trans. Systems Man Cybernet. 31 (2), 216–233.
Assabie, Y., Bigun, J., 2007. Multifont size-resilient recognition system for Ethiopic script. Internat. J. Document Anal. Recognition 10 (2), 85–100.
Assabie, Y., Bigun, J., 2008. Lexicon-based offline recognition of Amharic words in unconstrained handwritten text. In: The 19th Internat. Conf. Pattern Recognition (ICPR2008), December 8–11, Tampa, Florida, USA. IEEE.
Bar-Yosef, I., Hagbi, N., Kedem, K., Dinstein, I., 2009. Line segmentation for degraded handwritten historical documents. In: Proceedings of the 10th Internat. Conf. Document Analysis and Recognition (ICDAR2009), Barcelona, Spain, pp. 1161–1165.
Basu, M., 1994. Gaussian derivative model for edge enhancement. Pattern Recognition 27 (11), 1451–1461.
Bigun, J., 2006. Vision with Direction. Springer, Heidelberg.
Bigun, J., Granlund, G., 1987. Optimal orientation detection of linear symmetry. In: First International Conference on Computer Vision, ICCV, London, June 8–11. IEEE Computer Society, pp. 433–438.
Bigun, J., Bigun, T., Nilsson, K., 2004. Recognition by symmetry derivatives and the generalized structure tensor. IEEE TPAMI 26 (2), 1590–1605.
Bunke, H., 2003. Recognition of cursive Roman handwriting – past, present and future. In: Proc. 7th Internat. Conf. Document Analysis and Recognition, Edinburgh, pp. 448–459.
Cheriet, M., Kharma, N., Liu, C.-L., Suen, C., 2007. Character Recognition Systems. John Wiley, New York.

El-Yacoubi, A., Gilloux, M., Sabourin, R., Suen, C., 1999. An HMM-based approach for off-line unconstrained handwritten word modeling and recognition. IEEE TPAMI 21 (8), 752–760.

Fujisawa, H., 2008. Forty years of research in character and document recognition – An industrial perspective. Pattern Recognition 41 (8), 2435–2446.

Gerard, A., 1981. African language literatures: An introduction to the literary history of Sub-Saharan Africa. Three Continents Press, Washington.

Gordon, R., 2005. Ethnologue: Languages of the world, fifteenth ed. SIL International, Dallas.

Huang, C., Srihari, S., 2008. Word segmentation of off-line handwritten documents. In: Proc. Document Recognition and Retrieval XV, IST/SPIE Annual Symposium, vol. 6815.

Jain, A., Duin, R., Mao, J., 2000. Statistical pattern recognition: A review. IEEE TPAMI 22 (1), 4–37.

Khorsheed, M., 2003. Recognising handwritten Arabic manuscripts using a single hidden Markov model. Pattern Recognition Lett. 24 (3), 2235–2242.

Kim, K., Kim, D., Aggarwal, J., 1998. Feature extraction of edge by directional computation of gray-scale variation. In: Proc. 14th Internat. Conf. on Pattern Recognition (ICPR'98), vol. 2, pp. 1022–1027.

Koerich, A.L., Leydier, Y., Sabourin, R., Suen, C.Y., 2002. A hybrid large vocabulary handwritten word recognition system using neural networks with hidden Markov models. In: Proc. IWFHR2002, pp. 99–104.

Koerich, A.L., Sabourin, R., Suen, C.Y., 2003. Lexicon-driven HMM decoding for large vocabulary handwriting recognition with multiple character models. 6, 126–144.

Li, Y., Zheng, Y., Doermann, D., Jaeger, S., 2006. A new algorithm for detecting text line in handwritten documents. In: Proc. 10th IWFHR, La Baule, France, pp. 35–40.

Liu, C.-L., Fujisawa, H., 2008. Classification and Learning Methods for Character Recognition: Advances and Remaining Problems. Springer, Berlin. pp. 139–161.

Liu, Z.Q., Cai, J., Buse, R., 2003. Handwriting Recognition: Soft Computing and Probabilistic Approaches. Springer, Berlin.

Lorigo, L., Govindaraju, V., 2006. Offline Arabic handwritten word recognition: A survey. IEEE TPAMI 28 (5), 712–724.

Louloudis, G., Gatos, B., Pratikakis, I., Halatsis, C., 2009. Text line and word segmentation of handwritten documents. Pattern Recognition 42 (12), 3169–3183.

Madhvanath, S., Govindaraju, V., 2001. The role of holistic paradigms in handwritten word recognition. IEEE TPAMI 23 (2), 149–164.

Marinai, S., Gori, M., Soda, G., 2005. Artificial neural networks for document analysis and recognition. IEEE TPAMI 27 (1), 23–35.

Meshesha, M., Jawahar, C.V., 2005. Recognition of printed Amharic documents. In: Internat. Conf. Document Analysis and Recognition (ICDAR), pp. 784–788.

Mori, S., Suen, C., Yamamoto, K., 1992. Historical review of OCR research and development. Proc. IEEE 80 (7), 1029–1058.

Plamondon, R., Srihari, S., 2000. On-line and off-line handwriting recognition: A comprehensive survey. IEEE TPAMI 22 (1), 63–84.

Rabiner, L., 1989. A tutorial on hidden Markov models and selected applications in speech recognition. Proc. IEEE 77 (2), 257–286.

Rabiner, L., Juang, B., 1986. An introduction to hidden Markov models. IEEE Acoust. Speech Signal Process. Mag. 3 (1), 4–16.

Ruiz-Pinales, J., Jaime-Rivas, R., Castro-Bleda, M.J., 2007. Holistic cursive word recognition based on perceptual features. Pattern Recognition Lett. 28 (13), 1600–1609.

Selvi, S., Indira, K., 2005. A novel character segmentation algorithm for offline handwritten character recognition. In: Proc. 10th IWFHR, Mysore, India, pp. 462–468.

Shi, D., Damper, R.I., Gunn, S.R., 2003. Offline handwritten Chinese character recognition by radical decomposition. ACM Trans. Asian Lang. Inform. Process. 2 (1), 27–48.

Srihari, S.N., 1992. High-performance reading machines. Proc. IEEE 80 (7), 1120–1132.

Srihari, S.N., Hong, T., Srikantan, G., 1997. Machine-printed Japanese document recognition. Pattern Recognition 80 (8), 1301–1313.

Suen, C.Y., Mori, S., Kim, S.H., Leung, C.H., 2003. Analysis and recognition of Asian scripts – The state of the art. In: Proc. 7th Internat. Conf. Document Analysis and Recognition, Edinburgh, pp. 866–878.

Tan, T., Sullivan, G., Baker, K., 1996. Efficient image gradient-based object localization and recognition. In: Proc. IEEE Computer Society Conf. Computer Vision and Pattern Recognition (CVPR'96), pp. 397–402.

Uchida, S., Sakoe, H., 2003. Eigen-deformations for elastic matching based handwritten character recognition. Pattern Recognition 36 (9), 2031–2040.

Vinciarelli, A., Bengio, S., Bunke, H., 2004. Offline recognition of unconstrained handwritten texts using HMMs and statistical language models. IEEE TPAMI 26 (6), 709–720.

Young, S., Evermann, G., Gales, M., Hain, T., Kershaw, D., Liu, X., Moore, G., Odell, J., Ollason, D., Povey, D., Valtchev, V., Woodland, P., 2006. The HTK Book. Cambridge University Engineering Department, Cambridge.