IJDAR
DOI 10.1007/s10032-007-0048-y

ORIGINAL PAPER

Multifont size-resilient recognition system for Ethiopic script

Yaregal Assabie · Josef Bigun

Received: 28 March 2006 / Accepted: 7 May 2007

Abstract This paper presents a novel framework for recognition of Ethiopic characters using structural and syntactic techniques. Graphically complex characters are represented by the spatial relationships of less complex primitives which form a unique set of patterns for each character. The spatial relationship is represented by a special tree structure which is also used to generate string patterns of primitives. Recognition is then achieved by matching the generated string pattern against each pattern in the alphabet knowledge-base built for this purpose. The recognition system tolerates variations on the parameters of characters like font type, size and style. Direction field tensor is used as a tool to extract structural features.

Keywords Optical character recognition · Ethiopic · Multifont · Structural and syntactic techniques · Direction field tensor

1 Introduction

Ethiopia has one of the longest continuous literature traditions. Ethiopic script has been used in the country as a unique writing system since the fifth century BC [18]. Ethiopic is a general term coined for Ethiopian Semitic languages, such as Amharic, Geez, Tigrigna, Gurageggna, etc. With the introduction of Christianity in the fourth century AD, the script was largely used by Geez, which was by then the official language of both the imperial court and the church. Since the fourteenth century, the official status of Geez in the imperial court has been slowly replaced by Amharic language which is itself derived from Geez [5]. Amharic remained the official language of the present day Ethiopia and the language grew as the second most spoken Semitic language in the world next to Arabic. However, Geez is also still serving as the liturgical language of Ethiopian Orthodox Church. There are also about 72 other languages spoken in Ethiopia and as a result the Ethiopic alphabet has evolved through many forms over centuries to better suit the vocal property of different languages. At present, the alphabet is most widely used by Amharic and a total of over 80 million people inside as well as outside Ethiopia are using the Ethiopic alphabet for writing.

Automatic character recognition is one of the early studied fields of pattern recognition problems. It is motivated by the need for automatic processing of large volumes of data in postal code reading, office automation, bank checks, and other business and scientific applications [25]. The history of optical character recognition (OCR) starts with the advent of computers and research on character recognition of Latin script for machine-printed text has been on the record since the early 1950s [24,25]. To recognize characters, different pattern recognition techniques like template matching, syntactic and structural, statistical and neural network approaches have been used.

Template matching is one of the simplest and earliest approaches of pattern recognition techniques where the charcater to be recognized is matched against a database of stored templates of characters [19,22,25]. Syntactic and structural techniques are used for recognition of complex patterns which are represented in terms of the interrelationships between simple sub-patterns called *primitives* [4,9,22,24]. Statistical approach is the most intensively studied technique which represents each pattern in terms of features or measurements [4,22,33], and many OCR systems make use

Y. Assabie (✉) · J. Bigun
School of Information Science,
Computer and Electrical Engineering,
Halmstad University, 301 18 Halmstad, Sweden
e-mail: Yaregal.Assabie@ide.hh.se

J. Bigun
e-mail: Josef.Bigun@ide.hh.se

of this technique. In comparison, the neural networks are recently developed pattern recognition techniques inspired by neuronal oparations in biological systems [4,17]. They are known to be more effective on handwritten character recognition [3,10,22]. Each of the recognition techniques have their own advantages and limitations, and hybrid systems draw upon the synergy effect of two or more techniques [4,9,21,22,30].

Most character recognition techniques pass through four stages: *preprocessing, segmentation*, *feature extraction*, and *classification*. Preprocessing includes image enhancement, noise removal, skewness correction, size normalization, and thinning [3,16]. These techniques are mostly independent of the scripts. Thus, preprocessing algorithms developed for one script can be normally adopted to another script. Segmentation is the step in which observed patterns in the image are segregated into units of patterns that seem to form characters [16,25]. Feature extraction involves the measurement or computation of the most relevant information for classification purpose. It is an important factor in achieving high recognition performance. The selection of discriminating features mostly depends on the nature of character structures and writing styles. Thus, features used for recognition of one script and writing style may not be applicable to another [16,17,33]. Classification is the final stage in character recognition in which each character is assigned to a certain class [25].

Intensive research has been carried out on recognition of machine-printed Latin characters and the research is now directed towards handwritten and cursive text recognition [10]. However, due to variations in the structure of scripts, the available recognition algorithms for Latin script could not be directly adopted for other scripts. Thus, OCR technology of non-Latin scripts generally lag behind the status of Latin OCR [31,32]. However, over the past few years, there has also been developments in the recognition of Kanji, Chinese, Arabic, Indian and other oriental scripts [1,15,19, 23–25,27,30–32].

Despite the large population that uses the alphabet, Ethiopic OCR technology is far behind the technologies developed for many largely used scripts, e.g., Latin, Chinese and Arabic. To the knowledge of authors, there are few published studies on Ethiopic character recognition. The methods used for recognition were template matching by taking signature property [13] and linear symmetry property [27] of characters. In this paper, we present a multifont, and size-resilient Ethiopic character recognition system. We propose structural and syntactic techniques for recognition of Ethiopic characters where the graphically complex characters are represented by less complex primitive structures and their spatial interrelationships. To this end, a special tree structure is developed in which primitives and their interrelationships are stored and a unique set of string patterns is generated for each character. Recognition is made by matching the generated patterns with each pattern in a stored knowledge base of characters. The recognition system does not need size normalization, thinning or other preprocessing procedures. The only parameter that needs to be adjusted during the recognition process is the size of a Gaussian window which should be chosen optimally in relation to font sizes. We constructed an Ethiopic Document Image Database (EDIDB) from real-life documents and the recognition system is tested by documents from the database.

The organization of the remaining sections of this paper is as follows. In Sect. 2, we present the structural analysis of Ethiopic alphabet along with primitive structures and their spatial interrelationships. Section 3 describes the representation of the interrelationships of primitives. Extraction of structural features is discussed in Sect. 4, and in this section, we also introduced direction field tensor which is used as a tool in the recognition process. In Sect. 5, we present the overall recognition system and experimental results are reported in Sect. 6. We conclude the paper in Sect. 7 and briefly discuss the future work. The relevant details of EDIDB are annexed at the end.

2 Structural analysis of Ethiopic script

The recently standardized set of Ethiopic alphabet has 435 characters. Roughly half of this set (238) of them are used by Amharic and considered as the most commonly used even in other Ethiopic languages. This subset of the Ethiopic alphabet is conveniently written in a tabular format of seven columns as shown in Table 1. The seven columns represent the vocal sounds of characters in the order of *ä*, *u*, *i*, *a*, *e*,*ə*, and *o*.

Table 1 Part of the Ethiopic alphabet

Base Sound	Orders						
	1st (ä)	2nd (u)	3rd (i)	4th (a)	5th (e)	6th (ə)	7th (o)
h	ሀ	ሁ	ሂ	ሃ	ሄ	ህ	ሆ
l	ለ	ሉ	ሊ	ላ	ሌ	ል	ሎ
h	ሐ	ሑ	ሒ	ሓ	ሔ	ሕ	ሖ
m	መ	ሙ	ሚ	ማ	ሜ	ም	ሞ
s	ሠ	ሡ	ሢ	ሣ	ሤ	ሥ	ሦ
r	ረ	ሩ	ሪ	ራ	ሬ	ር	ሮ
.	.	.	.	.	.	.	.
.	.	.	.	.	.	.	.
.	.	.	.	.	.	.	.
s'	ፀ	ፁ	ፂ	ፃ	ፄ	ፅ	ፆ
p'	ጰ	ጱ	ጲ	ጳ	ጴ	ጵ	ጶ
f	ፈ	ፉ	ፊ	ፋ	ፌ	ፍ	ፎ
p	ፐ	ፑ	ፒ	ፓ	ፔ	ፕ	ፖ
v	ቨ	ቩ	ቪ	ቫ	ቬ	ቭ	ቮ

The first column in each row (*ä* sound) represents *basic characters* and other columns show *modified characters*. Thus, Ethiopic script is a writing system in which vowel sounds (modifiers) are denoted by diacritical marks or other systematic modification of the basic characters. When a modifier is added to the base character, it can also change the original shape of the base character as in the case of ረ which can be modified to ር. Some modifiers do not add extra symbol and only change the shape of the base character, e.g., ለ which can be modified to ላ. This makes it difficult to use a model representing the basic characters for recognition of their derivatives.

Most of the modifiers of the basic characters follow a similar pattern across each column. The second (*u*) and third (*i*) orders are mostly formed by adding appendages at the middle and bottom of the right leg, respectively. Although formation of the fourth (*a*) order is irregular, it is mostly characterized by a shortened left leg of basic characters. The fifth (*e*) order is typically formed by adding a loop to the right leg usually at the bottom. The sixth (ə) order is marked by high irregularity. Some are formed by bending their legs and some looped characters add appendages at their loops. Others also add appendages to the left to form the sixth order. The seventh (*o*) order is formed by different ways. Most of the modifiers make the left leg of the basic character longer. Others form loops at the right top of the character and the rest show irregularity.

2.1 Primitive structures

The Ethiopic characters are considered to have the most attractive appearance when written with thick appendages, vertical and diagonal strokes, and thin horizontal lines. Thus, prominent structural features in the alphabet are appendages, vertical and diagonal lines. These structural features form a set of primitive structures in Ethiopic characters. In this study, we suggest seven primitive structures which are interconnected in different ways to form a character. Primitives differ from one another in their structure type, relative length, direction, and spatial position. The classes of primitives are given below.

1. *Long vertical line (LVL)*. A vertical line that runs from the top to bottom level of the character. The primitive is found in characters like ሀ, በ, and ዘ.
2. *Medium vertical line (MVL)*. A vertical line that touches either the top or the bottom level (but not both) of a character. Characters like ሰ, ከ, and ና have these primitives.
3. *Short vertical line (SVL)*. A vertical line that touches neither the top nor the bottom level of the character. It exists in characters like ሐ, ቀ, and ሶ.
4. *Long forward slash (LFS)*. A forward slash primitive that runs from the top to the bottom level of a character. It is found in few characters like ሂ, ሥ, and ፖ.
5. *Medium forward slash (MFS)*. A forward slash primitive that touches either the top or the bottom level (but not both) of a character. Examples are አ, ፉ, and ነ.
6. *Backslash*. A line that deviates from the vertical line position to the right when followed from top to bottom. The characters ል, ስ, and ሽ have such primitives.
7. *Appendages*. Structures which have almost the same width and height. These primitives are found in many characters. Examples are ሂ, ት, and ቸ.

2.2 Spatial interrelationships between primitives

The unique structure of characters is determined by primitives and their interconnection. The interconnection between primitives describes their spatial relationship. A primitive structure can be connected to another at one or more of the following regions of the structure: top (t), middle (m) , and bottom (b). There can be one, two or three connections between two primitives. The first connection detected as one goes from top to bottom is considered as the *principal connection*. Other additional connections, if they exist, are considered as *supplementary connections*. The spatial relationship between two primitives α and β with a principal connection is represented by the pattern $\alpha\mathbf{z}\beta$, where **z** is an ordered pair (x,y) of the connection regions with x, y taking the values t, m, or b. In this pattern, α is connected to β at region x of α, and β is connected to α at region y of β. When two primitives are connected by more than one connector, e.g., in the character ቀ, where LVL and SVL are connected by two connectors, then supplementary connection exists between α and β, and the connection type **z** will be a set of ordered pairs of the connection regions. In the pattern $\alpha\mathbf{z}\beta$, primitive α is said to be spatially located to the left of β. Thus, the spatial relationship between two primitives is described in terms of *spatial position* (left or right) and *connection type*.

In this study, we reveal 18 connection types between two primitives. As shown in Table 2, there are nine principal

Table 2 Connection types between two primitives with example characters in brackets

Principal Connections	Supplementary Connections					
	None	(m,b)	(b,m)	(b,b)	(m,m) (b,m)	(m,m) (b,b)
(t,t)	በ(በ)	ቦ(ቦ)	ቧ(ቧ)	ቧ(በ)	ቧ(ቧ)	ፀ(ፀ)
(t,m)	ሐ(ሐ)		ቀ(ቀ)			
(t,b)	ሶ(ሰ)					
(m,t)	ከ(ሐ)	ቱ(ቀ)	ኳ(ፍ)	ኈ(ዚ)		
(m,m)	ዘ(ዘ)					
(m,b)	ሥ(ሠ)					
(b,t)	ጎ(ጎ)					
(b,m)	ሃ(ሃ)					
(b,b)	ሀ(ሀ)					

connections without supplementary connections, and nine principal with supplementary connections. The principal connection between two primitives is an ordered pair formed by the possible combinations of the three connection regions. This will lead to nine principal connections without supplementary connections as represented by the set: {(t,t),(t,m),(t,b),(m,t),(m,m),(m,b),(b,t),(b,m),(b,b)}. The principal connection (t,t), i.e., two primitives both connected at the top, has five types of supplementary connections: {(m,b),(b,m),(b,b),(m,m)+(b,m),(m,m)+(b,b)}. The principal connection (t,m) has only one supplementary connection: {(b,m)}. The principal connection (m,t) has three types of supplementary connections: {(m,b),(b,m),(b,b)}. The remaining principal connections do not have any supplementary connection.

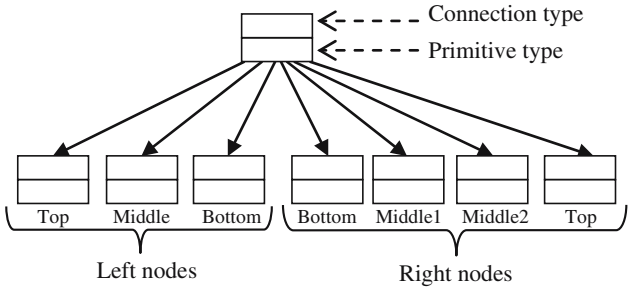

Fig. 1 General tree structure of characters

3 Representation

The structural analysis of Ethiopic characters in Sect. 2 shows that each character can be uniquely represented by a set of primitive structures and their spatial relationships. This invites to apply structural and syntactic techniques for recognition. In structural and syntactic pattern recognition systems, a pattern is represented by the synthesis information that combines sub-patterns to generate a complex pattern. Common symbolic data structures that can store this information are trees, graphs, and strings [4]. In each data structure, there is a tradeoff between the representation power and computational complexity. String matching has less computational complexity but the representation power is limited since strings cannot naturally be extended to multi-dimensional data. On the other hand, trees and graphs have better representational power for higher dimensional data but are costly for computation [4,11,28]. In this study, a tree data structure is used for representation of primitives and their spatial relationships whereas for computational processing of the data, the tree is converted to a string data structure. Thus, trees are used as intermediate tool to generate one-dimensional strings of primitives and their interrelationships for each Ethiopic character.

3.1 The proposed tree data structure

In Sect. 2.2, we discussed that there are three connection regions for a primitive: top, middle, and bottom. In Ethiopic script, to the left of a primitive structure, at most one primitive is connected at each connection region. For example, in the character ሸ, three appendage primitives are connected to the left of the Long Vertical Line primitive at each of the three connection regions. Therefore, a maximum of *three* primitives can be connected to the left of another primitive. In the above example, three appendages are also connected to the right of the Long Vertical Line primitive at the three connection regions. In addition, there are cases where two primitives are connected at the middle of a primitive as in ሐ and ሠ. Therefore, we have the possibility of having *four* connections to the right of a primitive although not at the same time. To represent this relationship, a special tree structure having three left nodes and four right nodes is proposed. A general representation of the tree is shown in Fig. 1. Each node in the tree stores data about the type of the primitive itself, the type of connections with its parent primitive, and the spatial positions of primitives connected to it.

A child primitive is appended to its parent primitive at one of the seven child nodes in the tree based on the principal connection that exists between them. The principal connection provides information about the spatial position of child primitives. The existence of supplementary connections does not have impact on the spatial position of the child primitive but only affects the connection type between the child and parent primitive. The connection types a parent can have with its child primitives at different spatial positions and connection regions is presented in Table 3. Connection type of NONE is used when there is a primitive without being connected to any other primitive. Such cases are mostly observed in degraded documents. In this case, the primitive is appended to one of other primitives based on the closeness in their spatial position. The connection type of the root primitive, which has not any parent primitive, is also NONE.

For implementation, a two digit numerical code is assigned for each primitive and their connection types. The first digit in the numerical code of primitives represents their relative length and/or structure, and the second digit represents their direction. Numerical code of primitives is presented in Table 4. In the case of connection types, the two digits represent left and right spatial positions, respectively. The three connection regions, i.e., top, middle, and bottom are represented by the numbers 1–3, respectively. The whole connection type is represented by concatenation of the numbers. For example, connections (t,m), (m,t)+(b,m), (t,t)+(m,m)+(b,m) are represented by 12, 2132, and 112232, respectively. A connection type of NONE is represented by 44.

Table 3 Connection types between parent and its child primitives at different positions

Left			Right		
Top	Middle	Bottom	Bottom	Middle1 & Middle 2	Top
(t,t)	(t,m)	(t,b)	(b,t)	(m,t)	(t,t)
(t,t)+(m,b)	(t,m)+(b,m)	(m,b)	(b,m)	(m,t)+(m,b)	(t,t)+(m,b)
(t,t)+(b,m)	(m,m)	(b,b)	(b,b)	(m,t)+(b,m)	(t,t)+(b,m)
(t,t)+ (b,b)	(b,m)			(m,t)+ (b,b)	(t,t)+ (b,b)
(t,t)+(m,m)+(b,m)				(m,m)	(t,t)+(m,m)+(b,m)
(t,t)+(m,m)+(b,b)				(m,b)	(t,t)+(m,m)+(b,b)
(m,t)					(t,m)
(m,t)+(m,b)					(t,m)+(b,m)
(m,t)+(b,m)					(t,b)
(m,t)+(b,b)					
(b,t)					

Table 4 Numerical codes assigned to primitive structures

Primitive types	Numerical codes
Long vertical line	98
Medium vertical line	88
Short vertical line	78
Long forward slash	99
Medium forward slash	89
Backslash	87
Appendages	68

3.2 Building primitive tree

The first step in building a primitive tree is identifying the root primitive. Variation in setting the root primitive results in different tree structures which will affect the pattern generated for a character. Thus, to build a consistent primitive tree structure for each character, a primitive which is spatially located at the left top position of the character is used as the root primitive.

There are few characters like ሰ where three primitives are interconnected to each other and results in one more connection than ordinary connections between primitives. In such cases, three primitives with their respective connection types are recorded once and a node that stores only the extra connection type will be added. Figure 2 illustrates primitive trees of characters having multiple connections between two primitives (ቀ), two connections at the right middle of a primitive (ከ), and three interconnected primitives (ሰ).

The following recursive algorithm is developed to build primitive tree of characters. The function is initially invoked by passing the root primitive as a parameter.

```
BuildPrimitiveTree (Primitive)
{
  BuildPrimitiveTree (LeftTopPrimitive)
  BuildPrimitiveTree (LeftMidPrimitive)
  BuildPrimitiveTree (LeftBotPrimitive)
  BuildPrimitiveTree (RightBotPrimitive)
  BuildPrimitiveTree (RightMid1Primitive)
  BuildPrimitiveTree (RightMid2Primitive)
  BuildPrimitiveTree (RightTopPrimitive)
}
```

4 Feature extraction using direction field tensor

In the previous section, we described that the characters are represented by patterns of less complex structural features. Primitives and connectors are the structural features used for recognition of characters. Thus, extraction of these structural features forms the basis of the recognition process. We use direction field tensor as a tool for extraction of the structural features. Direction field tensor, which is introduced by [8] for multi-dimensions, has been used in several pattern recognition problems over the past decade. Below we give a brief summary of the direction tensor, which is explained in detail in [6].

4.1 Direction field tensor

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. The direction tensor, also called the structure tensor [6,7,34], is a real valued triplet, which is a tensor representing the local directions of pixels. For a local neighborhood $f(x, y)$ of an image f, the direction tensor S is computed as a 2×2 symmetric matrix using Gaussian derivative operators D_x and D_y.

$$S = \begin{pmatrix} \iint (D_x f)^2 \mathrm{d}x\mathrm{d}y & \iint (D_x f)(D_y f)\mathrm{d}x\mathrm{d}y \\ \iint (D_x f)(D_y f)\mathrm{d}x\mathrm{d}y & \iint (D_y f)^2 \mathrm{d}x\mathrm{d}y \end{pmatrix} \quad (1)$$

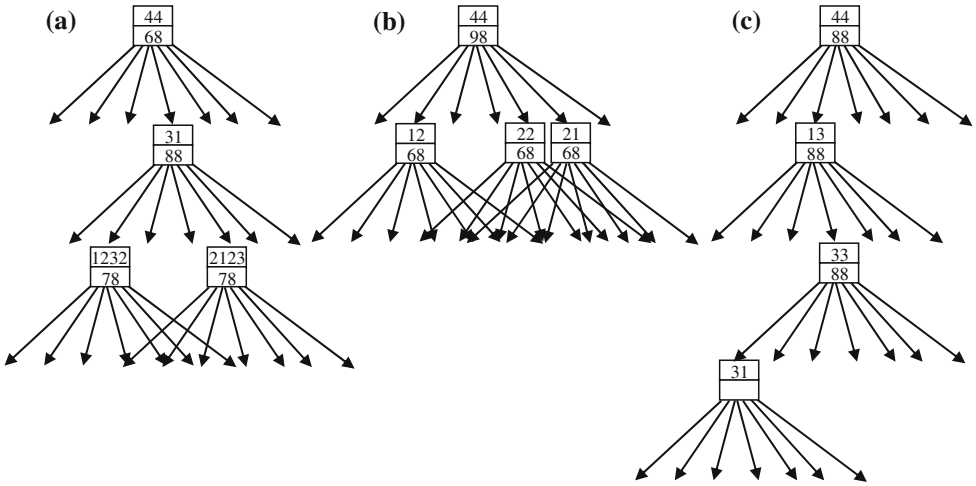

Fig. 2 Primitive trees for **a** ⚘, **b** ⚘, and **c** ⚘

The integrals can be implemented as convolutions with a Gaussian kernel which is isotropic and is defined as follows:

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

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

$$gx(x) = \frac{1}{\sqrt{2\pi}\sigma} \exp\left(-\frac{x^2}{2\sigma^2}\right) \tag{3}$$

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

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:

$$\mathrm{I20} = \iint ((D_x + \mathrm{i}D_y) f)^2 \mathrm{d}x\mathrm{d}y \tag{5}$$

$$I11 = \iint \left|(D_x + \mathrm{i}D_y) f\right|^2 \mathrm{d}x\mathrm{d}y \tag{6}$$

The complex partial derivative operator $D_x + \mathrm{i}D_y$ is defined as:

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

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 of S) and the magnitude is a measure of the local LS strength (the difference of eigenvalues of S). The scalar I_{11} measures the amount of gray value changes in a local neighborhood of pixels (the sum of eigenvalues). Direction field tensor, which is a tensor field defined over local images for all points in the entire image, can thus also be conveniently represented by the 2D complex I_{20} and 1D scalar I_{11}. In the implementation (using MATLAB), I_{20} and I_{11} are computed as follows:

1. Generate a 2D Gaussian kernel and a 2D Gaussian derivative kernel. Since Gaussian functions are separable, two 1D Gaussian kernels (*gx* and *gy*) and two 1D Gaussian derivative kernels (*dx* and *dy*) can be used for better efficiency.
2. Apply convolution operations on the original image img to generate *dxf* and *dyf* .

$$dxf = gy^* \left(dx^* \mathrm{img}\right)$$
$$dyf = gx^* \left(dy^* \mathrm{img}\right)$$

3. Compute:
   a. $\hat{\mathrm{I}}_{20}$ from *dxf* and *dyf* by pixel-wise complex squaring. $\hat{\mathrm{I}}_{20} = (dxf + j^* dyf)^2$ where $j = \sqrt{(-1)}$
   b. $\hat{\mathrm{I}}_{11}$ from abs($\hat{\mathrm{I}}_{20}$).
4. Compute the complex image I_{20} and the scalar image I_{11} from $\hat{\mathrm{I}}_{20}$ and $\hat{\mathrm{I}}_{11}$, respectively by averaging with a second set of Gaussian kernels.

The complex image I_{20} can be displayed in color as shown in Fig. 3b where the hue represents direction of pixels in double angle representation. Pixels with directions of zero are represented by red color. Another way of displaying the complex image is to make use of vectorial representation as shown in Fig. 3c, where the arrows show direction in double angle representation and the magnitude shows the LS strength of pixels.

4.2 Extraction of structural features

After computing the direction field tensor, extraction of structural features involves three steps: (1) classifying pixels (low-level features) as parts of the would-be primitives and connectors; (2) extracting skeletons of the I_{20} image and

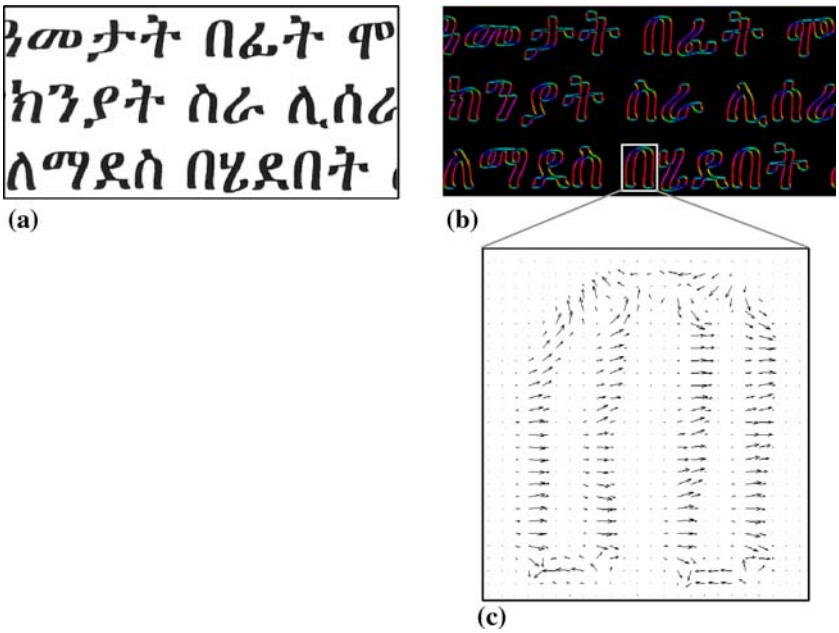

Fig. 3 **a** Ethiopic text, **b** I_{20} of **a**, **c** vectorial representation of I_{20}

forming lines (intermediate-level features) from pixels; and (3) extracting primitives and connectors (high-level features) from the skeletal form of the I_{20} image. These steps are discussed below including segmentation of characters which is done after skeletal lines are extracted in the I_{20} image.

4.2.1 Classifying pixels

The LS strength of I_{20} image is normalized to the range of [0,1]. In the normalized I_{20} image, pixels of text region can be selected by using the strength of LS property (ρ). In this study, pixels with $\rho \geq 0.05$ (strong LS property) are considered as parts of primitive and connectors of characters. Since the direction of pixels is represented by double angle, the angle θ obtained from the argument of I_{20} is in the range of 0–180°. The direction of pixels at the edges of primitives is close to 0–180° and can be converted to the range of 0–90° by $\varepsilon = \text{abs}(90 - \theta)$, so that ε for primitives is consistently close to 90°. Among the pixels with strong LS property, those with $\varepsilon \geq 30°$ are considered as parts of primitives, and others are considered as parts of connectors. Then, structural features are further extracted as shown in Fig. 4.

4.2.2 Extracting skeleton of the I_{20} Image

Due to the Gaussian filtering used in the computation of direction tensor, the LS strength at the orthogonal cross-section of edges forms a Gaussian of the same window size as the Gaussian filter. Pixels that form edges in the grayscale image correspond to the mean of the Gaussian at the orthogonal cross-section in the corresponding local neighborhood of the I_{20} image. Therefore, the cross-section of lines in the I_{20} image can be reduced to a skeletal form (one pixel size) by taking the point closest to the mean of the Gaussian formed by the LS strength in the orthogonal direction. To keep the continuity of skeletons, candidate pixels are taken from a local neighborhood of the head of skeleton. The skeletal lines in the I_{20} image form the contour lines of the original grayscale image. The skeletal form of the I_{20} image shown in Fig. 5.

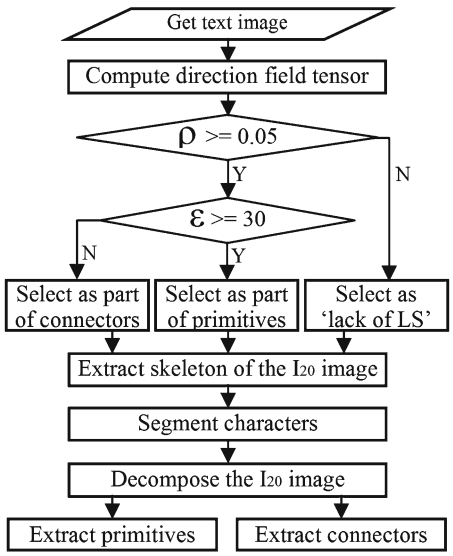

Fig. 4 Flowchart of structural features extraction

4.2.3 Segmenting characters

Text lines in the I_{20} image show strong LS property, whereas the horizontal spaces between text lines are characterized

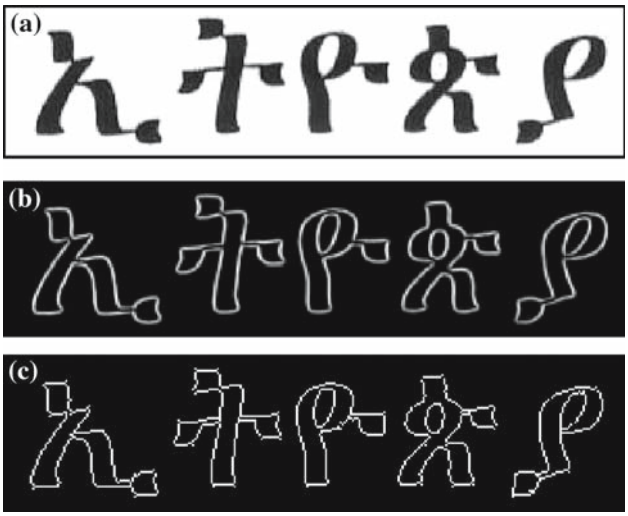

Fig. 5 **a** Ethiopic text, **b** LS strength of I_{20} of **a**, and **c** skeletal form of **b** without direction information

Fig. 7 **a** Ethiopic text, **b** primitive lines shown as *white lines*, and connectors shown as the *gray lines*, **c** extracted primitive structures

by lack of LS property. Therefore, text lines are segmented in the skeletal form of the I_{20} image by searching for areas that lack LS property. For each segmented text line, vertical spaces that lack LS are used to segment each character in the line. Figure 6 shows segmented characters in rectangular boxes.

4.2.4 Extracting primitives and connectors

For each segmented character, the skeletal form of I_{20} image can be decomposed in to two images of primitives and connectors by using the direction and LS strength of pixels as described before. A linear structure in the gray scale image forms two lines in the I_{20} image. The lines are formed as a result of extracting the skeleton from the I_{20} image. Connectors are made up of two matching horizontal lines (top and bottom) in the skeletal image. A primitive structure in the grayscale image will also have two vertical/diagonal lines (left and right edges) in the skeletal form of the I_{20} image. Primitive structures are then constructed by the two matching left and right lines. The group information about direction and spatial position of pixels in a primitive are finally used to classify the primitives. Figure 7 shows extracted primitives from the left and right edges of the text.

4.2.5 Restoring broken primitives

Primitive lines which make up primitive structures can be broken due to image noise, degraded document, misclassification of some parts of primitive lines as connectors, and junction with connectors. When only one of the primitive lines is broken, it can be restored by following the other matching primitive line as shown in Fig. 8. In Fig. 8c, the left and right primitive lines of the longest primitive structure in the character ም are broken due to junction and noise, respectively. The junction created lack of LS at the left edge of primitive structure, resulting in two left primitive lines. The continuity of the two left primitive lines, which are part of one long left primitive line, is restored by following the corresponding area of the right primitive line as shown in Fig. 8d.

When a primitive structure is connected to two other primitive structures both at the same area, the left and right primitive lines are broken resulting in two or more broken primitive structures. Under normal circumstances, there are connector lines at the top and bottom of primitive structures resulting in closed ends. When a primitive structure is broken due to junctions, connector lines will not exist due to lack of LS. Therefore, the broken primitive structures are left with open ends. In this case, the primitive structure can be restored by merging the broken primitive structures with open ends as shown in Fig. 9.

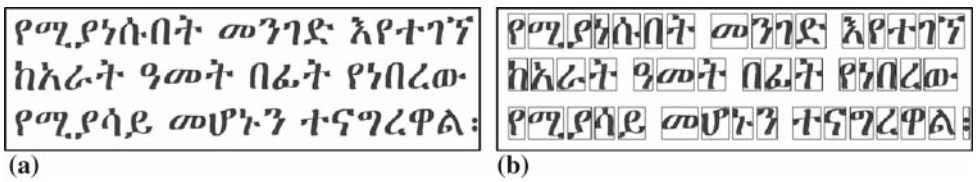

Fig. 6 **a** Original Ethiopic text, **b** segmented characters

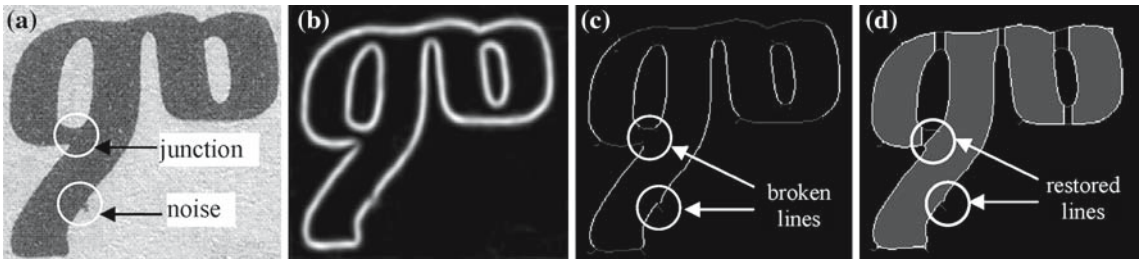

Fig. 8 **a** The character ፱ with noise, **b** LS strength I_{20} of **a**, **c** connector and primitive lines extracted from skeletal form of **b**, **d** extracted primitive structures of **a**

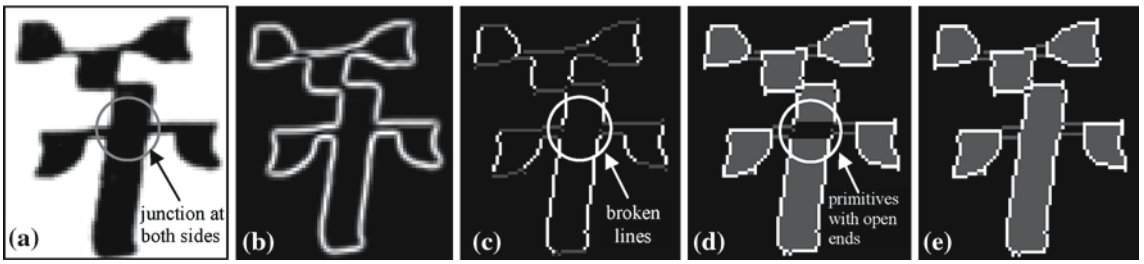

Fig. 9 **a** The character ፕ, **b** LS strength of I_{20} of **a**, **c** connector and primitive lines extracted from skeletal form of **b**, **d** extracted primitive structures of **a**, **e** restored primitive structures

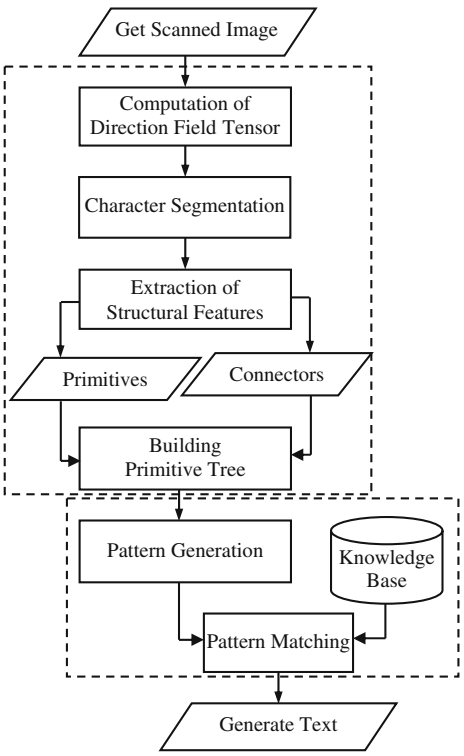

Fig. 10 Flowchart of the recognition process

5 The recognition system

The recognition system takes the document image as an input and generates its equivalent text as an output. As shown in Fig. 10, it can be seen as having two stages: (1) representation of characters in a tree structure of primitives and their spatial relationships, and (2) recognizing characters using the tree representation. The first stage is discussed in the previous sections, and the second stage of the recognition system is explained below.

5.1 Pattern generation

Trees provide a convenient representation for handling the spatial interrelationships of primitives. They are, however, costly in terms of further processing [4,11]. In some characters, there are primitives which compete closely to be selected as root primitives for the tree. As a consequence, there is a probability that more than one type of tree structure be built for a character. This means computational efficiency is an issue since significant comparison processing and storage will be needed. String patterns are more efficient than trees for computation purpose, and therefore, we convert primitive tree patterns to their corresponding string patterns for further processing.

In the domain of data structure, binary search trees provide an efficient way of storing data in an orderly manner. In binary search trees, the values of all child nodes in the left subtree of a given node are less than the value of the parent node, and the values of all child nodes in the right subtree of a given node are greater than the value of the parent node [29]. Data stored in a binary search tree can be traversed and converted to a string pattern in three ways: pre-order (parent–left–right), in-order (left–parent–right) and post-order (left–right–parent). In such trees, in-order traversal generates data

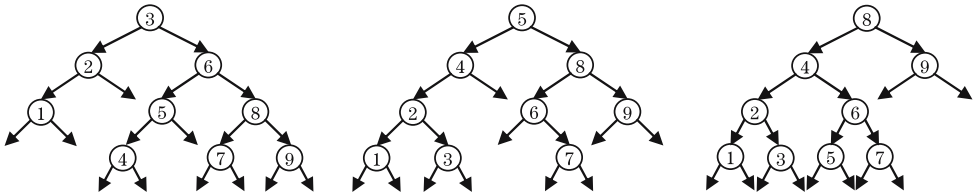

Fig. 11 Binary search trees for the same set of data with different root nodes

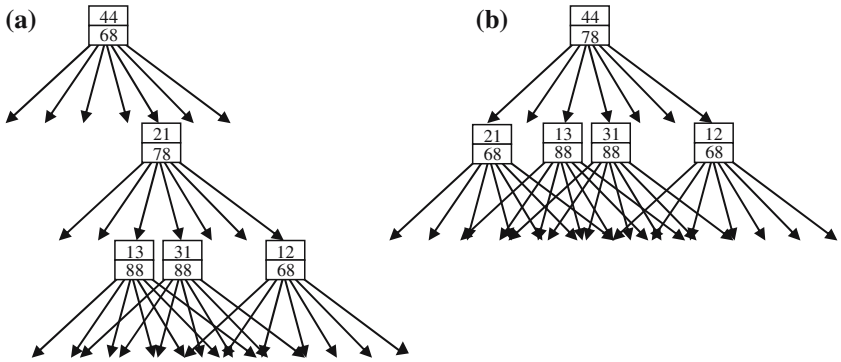

Fig. 12 Primitive trees for ﻫ with root primitive **a** ﹷ, **b** ﹹ

in ascending order whereas pre-order and post-order traversals generate them neither in ascending nor in descending order. It means that in-order traversal uniquely generates the same string pattern out of different binary search trees that are built from the same set of data [29]. This is because, for a given set of data, their arrangement in an ascending order forms a unique pattern no matter how they are arranged in the binary search tree. For example, in Fig. 11, the three binary search trees are built from the same set of data but different root nodes. The pattern generated by in-order traversal of the three trees is the same: {1, 2, 3, 4, 5, 6, 7, 8, 9}.

Our primitive tree behaves essentially as a binary search tree in the sense that primitives are systematically appended to other primitives in an orderly manner with respect to their relative spatial position and relationships. Consider primitives α, β, χ, δ, and γ. Then α is said to be less than γ if α is connected to the left side of γ. We also say $\alpha < \beta < \chi$ if they are connected to the left of γ at the top, middle and bottom positions, respectively. On the other hand, $\alpha < \beta < \delta < \chi$ if they are connected to the right of γ at the bottom, middle1, middle2, and top positions, respectively. The primitive tree is built with such orderly manner. We can convert each primitive tree to a string by using *in-order traversal* of the tree (*left*{top, middle, bottom}, *parent, right*{bottom, middle1, middle2, top}). In-order traversal of the tree keeps the relative order of spatial relationship and position of primitives in the same way as the binary search trees do. For a given character, this traversal also consistently generates the same pattern of primitives from trees that are built differently because of variations in selecting the root primitive. Figure 12 shows the character represented by two different trees as a result of selecting different root primitives which are spatially located at the left top of the character.

In-order traversal of the tree in Fig. 12a generates the pattern {44, **68**, 13, **88**, 21, **78**, 31, **88**, 12, **68**} and Fig. 12b generates {21, **68**, 13, **88**, 44, **78**, 31, **88**, 12, **68**}. The order of primitives in the two string patterns is the same. The order of connection types of the two strings can be made the same by swapping the connection type of the root primitive in the string pattern of Fig. 12b, which is always 44, with the first connection type in the string. This rule also works for other character strings. Thus, with minimal computation, a character with no major structural differences in its fonts can be uniquely represented by a string. The proposed in-order traversal algorithm is implemented using a recursive function in a similar way as building the primitive tree. This function is also invoked by passing the root primitive as a parameter.

```
GeneratePattern (Primitive)
{
 GeneratePattern (LeftTopPrimitive)
 GeneratePattern (LeftMidPrimitive)
 GeneratePattern (LeftBotPrimitive)
 PrintPattern (ConnectionType, PrimitiveType)
 GeneratePattern (RightBotPrimitive)
 GeneratePattern (RightMid1Primitive)
 GeneratePattern (RightMid2Primitive)
 GeneratePattern (RightTopPrimitive)
}
```

5.2 Alphabet knowledge base

The geometric structures of primitives and their spatial relationships remain the same under variations in fonts and their sizes. In Fig. 13a, all the different font types and sizes of the character ﻝ are described as two Long Vertical Lines both connected at the top. This is represented by the pattern {44, 98, 11, 98}. In such patterns, only the relative size of

Fig. 13 **a** The character ሀ with different types and sizes of 12 and 18, **b** bold style of **a** and **c** italic style of **a**

Table 5 Part of the alphabet knowledge base

Char	C1	P1	C2	P2	C3	P3	C4	P4	.	.	.
ሀ	44	98	33	98							
ሁ	44	99	33	99							
ሁ-	44	98	33	98	22	68					
ሁ-	44	99	33	99	22	68					
ሃ	44	98	11	98							
ሃ-	44	98	11	98	22	68					
ሄ	44	88	1132	98	11	88	1133	88			
ሂ	44	98	22	98							
ሇ	44	68	11	98	22	98					
.											
.											
.											

primitives, not the absolute size, is encoded. As there is no structural difference between a long vertical line and its bold version, Fig. 13b is also represented by the same pattern as in the case of Fig. 13a. In Fig. 13c, the character is described as two Long Forward Slashes both connected at the top and it is represented by the pattern {44, 99, 11, 99}. Therefore, any form of the character is represented as a set of patterns {{44, 98, 11, 98}, {44, 99, 11, 99}}.

Accordingly, the knowledge base of the alphabet consists of a set of possibly occurring patterns of primitives and their relationships for each character. This makes the proposed recognition technique tolerant of variations in the parameters of fonts. Table 5 shows part of the knowledge base for some characters with different fonts and writing styles. In the table C1, C2, C3, . . . , etc. indicate connection types whereas P1, P2, P3, . . . , etc. indicate primitives.

5.3 Pattern matching

Pattern matching is a fundamental step in the recognition process in which unknown input strings are compared against each pattern in the knowledge base. Several pattern matching algorithms have been introduced over the past decades for general pattern recognition problems [20]. Although the algorithms are usually customized to specific problems, pattern matching algorithms are classified in two broad categories: *exact* and *approximate* pattern matching algorithms. Exact pattern matching algorithms find an exact occurrence of input pattern in the knowledge base whereas approximate pattern matching algorithms find the occurrences of an input pattern within some threshold of error. Approximate pattern matching measures the similarity of two strings and the similarity measure can be computed by using distance functions [20,26].

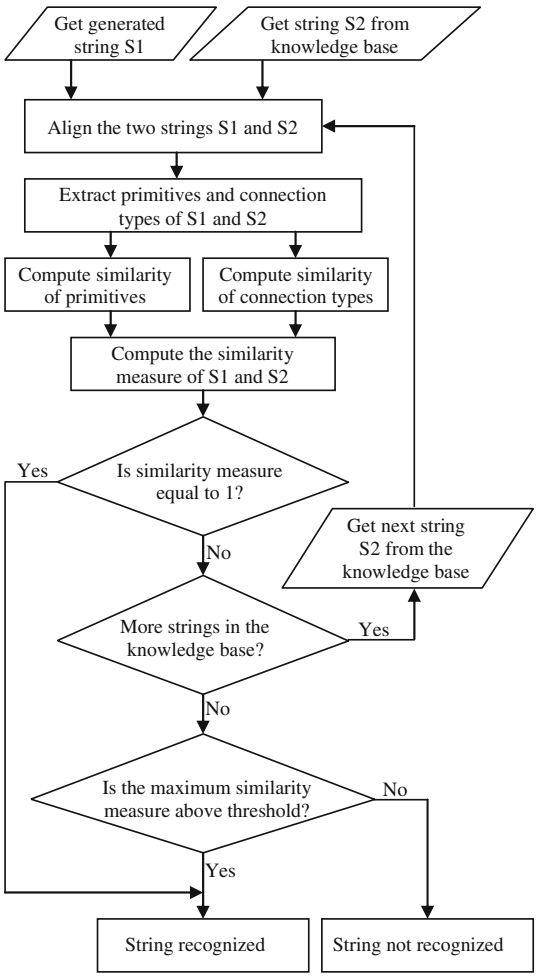

Fig. 14 Pattern matching algorithm

Extraction of primitives and their spatial relationships is sometimes affected by noise that may exist in the original image. Thus, it is not always possible to generate the desired string pattern for a character. To take the advantage of tolerating such small errors, approximate pattern matching technique is employed for recognition of strings. Figure 14 depicts the algorithm used for deciding whether an unknown input string is recognized or not, based on the maximum similarity measure of the string against with each string in the knowledge base. The two strings considered for comparison are aligned optimally based on the similarity of primitives. Once strings are aligned, the overall similarity measure is computed as a cumulative effect of the similarity of the corresponding individual components of the strings.

Fig. 15 Sample documents from EDIDB showing **a** Visual Geez 2000 Main font, **b** Visual Geez 2000 Agazian font, **c** Visual Geez Unicode font with italic style, **d** book, **e** newspaper, **f** magazine

The string size of each primitive type is 2. The similarity P of two primitives α and β with numerical codes $\alpha_1\alpha_2$ and $\beta_1\beta_2$ is computed as:

$$P = \frac{6 - (|\alpha_1 - \beta_1| + |\alpha_2 - \beta_2|)}{6}$$

Whenever the numbers of primitives in the two character strings differ, an empty primitive is used as a place holder, and the overall similarity value of any primitive with an empty primitive is set to be 0. This keeps the similarity value P between two primitives to be in the range between 0 and 1, with higher value showing better similarity.

The string size of a particular connection type ranges from 2 (only one connection) to 6 (three connections). Comparison between two connection types is made first by optimally aligning the substrings. Then, the similarity C of two connection types χ and δ is computed as

$$C = \frac{6 - \left(\sum_{i=1}^{n} \frac{|\chi_i - \delta_i|}{n}\right)}{6},$$

where n is the size of the longest substring. Whenever the two substrings do not match in their length, a connection type of NONE (numerical code 44) is inserted in the vacant places. To make the measurement consistent, the difference between an empty connection (NONE) and any other connection type is considered to be 6. Thus, the similarity value C between two connection types ranges from 0 to 1, with higher value for better similarity.

The overall similarity measure K between two character strings is the mean value of the similarity between corresponding components, i.e., primitives and connection types. Therefore, K is computed as follows:

$$K = \frac{\sum_{j=1}^{m} (C_j + P_j)}{2m},$$

where m is the total number of primitives in the longest character string. The value of K is in the range of [0,1]. For two strings having the same pattern, K has a value of 1 and the value decreases to 0 as the difference between two strings increases. Whether or not two strings are considered similar is determined by a threshold. A character is then said to be recognized when the most similar pattern in the knowledge base has a similarity measure of above the threshold value. To balance false acceptance and false rejection rates, a threshold value of 0.75 is used in this experiment. It means that, on average, for a pair of connection types and primitives $\chi\alpha$ of one string and $\delta\beta$ of another string, the following conditions, the thresholds of which are found empirically, are tolerated.

- χ and δ are the same ($C = 1$), α and β are different in orientation or relative size but not both ($P \geq 0.5$)
- at least one connection region of χ and δ are the same ($C \geq 0.5$), α and β are the same ($P = 1$)
- the difference between χ and δ is only one connection region ($C \geq 0.67$), α and β are close in orientation and the same in relative size, or vice versa ($P \geq 0.83$).

6 Experiments

6.1 Database development

A standard image database of Ethiopic text is not available so far for testing character recognition systems. Thus, we developed Ethiopic document image database (EDIDB) for testing the recognition system with different real life documents. EDIDB also helps to standardize the evaluation of Ethiopic character recognition systems in general. The database consists of wide range of document types taken from printouts, books, newspapers, and magazines. Specific details of the database are presented in Appendix A. The recognition system is tested with documents taken from EDIDB. Sample documents are shown in Fig. 15 and experimental results are discussed below.

6.2 Result

The robustness of the recognition system on variations in font type, size, style, document skewness, and document type is tested and results are presented below.

Table 6 Recognition results with different font types

Font type	Recognition (%)
Visual Geez Unicode	94
Visual Geez 2000 Main	94
Visual Geez 2000 Agazian	94
PowerGeez	94
GeezType	93

- *Font type*: Test documents have the same text content and font size of 12, but different font types. A total of 175 pages (about 35 pages for each font type) are used for testing the recognition result is presented in Table 6.
- *Font size*: Test documents have the same content of text with Visual Geez Unicode font but with different sizes. A total of 107 pages are used for testing (excluding font size 12, which is already tested above). The result is presented in Table 7.
- *Font style*: Test documents have the same content of text with Visual Geez Unicode font and 12 font size, but different formatting styles. About 70 pages, excluding the normal style which is already tested above, are used for testing and the result is presented in Table 8.
- *Skewness*: About 24 skewed documents are used to test the robustness of the recognition system with regard to skewness. The skewed documents are the same in content as that of Visual Geez Unicode font with 12 font size, and a comparison of the results is shown in Table 9.
- *Document type*: Test documents are taken from five books (a total of 48 pages), three newspapers (equivalent to 26 pages on A4 size paper), and two magazines (equivalent to 23 pages on A4 size paper). Results are summarized in Table 10.

Table 7 Recognition results with different font sizes

Font size	Recognition (%)
8	91
10	93
12	94
16	95
20	96

Table 8 Recognition results with different font styles

Font Style	Recognition (%)
Normal	94
Bold	95
Italic	92

Table 9 Recognition results with different document skewness

Skewness in degree	Recognition (%)
−20 and +20	88
−10 and +10	91
−5 and +5	94
0	94

Table 10 Recognition results with different document types

Document type	Recognition (%)
Books	89
Newspapers	86
Magazines	88

6.3 Discussion

The recognition system was tested on various real life Ethiopic documents of Amharic and Geez languages. The only parameter that is changed with variation in documents is the size of Gaussian window. The results obtained by Gaussian filtering operations are generally maximal when the window is perfectly symmetric along pixels. Thus, a Gaussian window size of even numbers is avoided in this experiment. The size of the Gaussian window is determined by the noise level of the document and the size of fonts. For clean documents, we used a window of 3×3 pixels for texts with font sizes of less than or equal to 12, a window of 5×5 pixels for font sizes of 16, and a window of 7×7 pixels for font sizes of 20. On the other hand, for most documents taken from books, newspapers, and magazines (font sizes are equivalent to about 12), a window of 5×5 pixels was used because of their higher level of noise. For noisy documents, a 5×5 pixels window sometimes over-smoothes characters. However, it still gives better recognition result than a 3×3 pixels window which is, as expected, not sufficient to remove excessive noises.

The major sources of errors in the recognition system are errors that arise from character segmentation and extraction of structural features. Character segmentation errors depend mainly on the quality of documents. In poor quality documents, thin horizontal lines in characters are easily lost during smoothing and give way for erroneous segmentation of a character into two or more parts. The noise in documents also hinders segmentation of characters. Although the segmentation accuracy of characters varies as such with the noise level and quality of documents, an average segmentation accuracy of 94% was achieved for books, newspapers, and magazines. For clean printouts which generally have less noise, about 98% of the characters were successfully segmented. Extraction of structural features is also influenced by font size and quality of documents. In the case of small font sizes, appendages and other small primitive structures are more

vulnerable to being lost due to over-smoothing. Noises that are close to edges of characters may also disrupt the process of structural feature extraction. The evaluation of extraction of structural features was made by automatically marking the extracted features (see Figs. 7–9) on the image and comparing with knowledge about the structural features that make up the characters. For documents taken from books, newspapers and magazines, an average of 93% of the primitive structures were successfully extracted. For clean printouts with font sizes of 16 and 20 about 99% of the primitive structures, for font size of 12 about 97% of the primitives structures, and for font sizes of 8 and 10 an average of 95% of primitive structures were successfully extracted.

Recognition of characters also depends on the nature of characters themselves. Characters that are formed out of more complex interrelation of smaller primitive structures like ሯ, ጅ and ፴ are recognized less efficiently. On the other hand, characters which are formed from simpler relationships of large primitive structure like በ, ዐ, ዘ and ጠ are recognized easily. Some groups of characters are also confused between each other than others. This includes ሀ and ሀ; ለ,ሰ and ሰ; ሸ and ሽ; ጥ, ፕ and ኘ; and ጎ and ጐ.

In general, character recognition results show that the recognition system can be used for various documents. The variations in the results are mainly due to the inherent characteristics of documents like noise level, which are also the case in Latin OCR softwares. For example, some clean pages from books, newspapers, and magazines were recognized in the same rate as clean printouts and other poor quality pages were recognized below the average result. For the case of smaller font sizes, we can improve the recognition accuracy by scanning documents at a higher resolution, which helps to increase the size of small primitive structures.

6.4 Comparison to other character recognition approaches

In Sect. 1, we mentioned some major approaches to solve pattern recognition problems: template matching; syntactic and structural; statistical and neural network classifiers. Every approach has its own advantages and limitations, and the approaches are not necessarily independent. Moreover, it is widely accepted that no single approach is best to solve all types of general pattern recognition problems.

As compared to other recognition approaches, our proposed system has an advantage of being resilient to variations in the characteristics of documents, which is a major issue of OCR systems. Template matching approach is the easiest method for implementation of recognition systems. However, it is computationally demanding. Templates made from the signature property of characters improved the processing speed, but the recognition was marked by high degree of confusion between Ethiopic characters [13]. Another drawback of template matching approach is its limitation to recognize various fonts. This requires storing the templates of characters with various font types and styles which is practically difficult, if not impossible, to exhaustively include all the variants. The processing speed becomes even worse as comparison is also to be made against each template. For Ethiopic script, a template matching of direction fields tested on a few set of characters with a specific font type, size, and style gives 92% accuracy [27]. However, this is not directly comparable to our results since the size and type of test data are different.

Statistical approaches describe characters in terms of d-dimensional feature spaces, and a suitable set of representative features with small intra-class and large inter-class variations must be selected. In Ethiopic script, since only small structures are added to modify the base characters, the modified characters retain the original shape of the base characters. This results in small inter-class variations which produces high degree of confusion even for template matching [13]. Consideration of various font types and styles would increase the intra-class variations which poses an additional problem. Moreover, since features are described as a fixed number of dimension, statistical approach usually requires preprocessing procedures like thinning, skew correction, size normalization. In fact, template matching also requires most of the preprocessing procedures. Our proposed approach does not require any of these preprocessing procedures, and in effect the total processing time is minimal. However, for Latin script, where the script is vowel-based (not modification based), the intra-class variation would be relatively larger. In fact, many Latin character recognition systems make use of the statistical approach. On top of this, linguistic models such as spell-checkers and part of speech analyzers are better studied for Latin languages, and used as post-processing tools to improve recognition results. With all these efforts, Latin OCR products suggesting a recognition rate of above 99% are now available on the market [12]. For multifont Chinese character recognition, Wu and Wu [35] used statistical techniques and an average recognition rate of 98% was achieved for practical Chinese documents with various fonts types. An average recognition rate of about 95% was also obtained for isolated Arabic characters with five different font types [2].

Neural networks are good classification tools if they are trained sufficiently. They have been effectively used for handwritten digit recognition, e.g., reading of zip codes. Traditionally, inputs to the neural network are pixel values of character images. Since the number of input nodes is a pre-determined fixed number, neural networks also require, among other procedures, normalization of the size of character images. Another drawback of neural network systems is that it is not possible to reason out how the result is obtained and therefore to introduce efficient remedy when they fail. The optimal

network model is obtained experimentally by the training and testing procedures [14].

6.5 Adaptability to other pattern recognition problems

Our approach offers a description of a complex pattern in terms of the relationships of its simpler sub-patterns. It can be adapted to other recognition problems provided that the complex patterns lend themselves to be described by spatial arrangements of simple sub-patterns. However, the specific representations of the relationships between sub-patterns vary according to the nature of the problem. Although it needs more study, we hope that the general framework will also be a useful basis for recognition of Ethiopic handwritten text.

The traditional approach for low level image processing is to make use of the gradient field. The direction field tensor, which we used for extraction of structural features, is more advantageous than the gradient field because it provides the optimal direction of pixels in a local neighborhood of an image in the total least square error sense. Direction field tensor also amplifies linear structures and suppresses non-linear structures. Therefore, it can be used as an efficient low-level image processing tool in a wide range of document processing and graphics recognition problems. Examples include recognition of engineering drawings, maps, diagrams, symbols, shapes, tables, and forms.

7 Conclusion and future work

In this paper, structural and syntactic techniques are applied to develop multifont and size-resilient recognition system for Ethiopic characters. Recognition of simpler structural features is easier than recognizing complex structures. Thus, we develop a recognition system by dealing with smaller constituents of character structures. Since the recognition system is insensitive to variations on font size, type and other parameters of characters, it does not require preprocessing techniques like thinning and size normalization. This, together with the 1D image processing that we use throughout, helps the system to be efficient in computational costs.

Images taken from clean printouts show better recognition performance due to their relatively better quality, and larger font sizes tend to be recognized only slightly better than their smaller versions. However, there was not a major difference in recognition accuracy due to variation in fonts. The recognition accuracy can still be improved by dedicating more efforts on character segmentation, extraction of structural features, knowledge base development and representation, and pattern matching algorithms. Extraction of structural features and pattern matching can be further improved by applying a hybrid system of neural networks and statistical techniques. The general framework of the proposed system can also be extended for recognition of handwritten characters and therefore, the research results are expected to lead to robust OCR systems for Ethiopic script.

Acknowledgments The authors would like to thank the Swedish International Development Agency (SIDA) for funding the study.

Appendix A: Ethiopic document image database for character recognition

I Introduction

With the growing need of computerized information processing, many researches in the area of recognition of various scripts have been conducted over the past decades. The types of documents used for testing recognition systems highly affect the results. Thus, to standardize and compare research results, text image databases are built for various scripts. Examples include NIST[1] for Latin, CEDAR[2] for Japanese, ERIM[3] for Arabic etc. However, there has no standard database for Ethiopic text so far. As part of the research on recognition of Ethiopic characters, we developed Ethiopic Document Image Database (EDIDB).[4] The database is also intended to serve other researchers in the area and standardize the research on Ethiopic character recognition. EDIDB helps researchers to test recognition systems with respect to variation on font type, font style, font size, document skewness, document uniformity and document type.

II EDIDB specification

Scanner: CanoScan LiDE 20 flatbed scanner
Resolution: 300 dpi
Image format: JPEG, Grayscale
Total Pages: 1,204 images
Languages: Amharic and Geez
Document types: Printouts, books, newspapers, and magazines

(1) Printouts
   a. Total scanned pages: 983
   b. *Fonts types*: Geez Type, Power Geez, Visual Geez 2000 Main, Visual Geez 2000 Agazian, and Visual Geez Unicode

[1] http://www.nist.gov/srd/optical.htm.

[2] http://www.cedar.buffalo.edu/Databases/JOCR/.

[3] http://documents.cfar.umd.edu/resources/database/ERIM_Arabic_DB.html.

[4] Researchers in the area of Ethiopic character recognition can contact the authors to get the database for free.

   c. *Font sizes*: 8, 10, 12, 16, 20
   d. *Font styles*: Normal, Italic, and Bold
   e. *Skewness*: Non-skewed, and skewed from −30° to 30°
   f. *Uniformity of documents*: Uniform in font size and style, combination of various font sizes (8–20) and styles (normal, bold, italic, and bold+italic)
(2) Books
   a. Total books: 5
   b. Total scanned pages: 116
(3) Newspapers
   a. Total newspapers: 3
   b. Total scanned pages: 79
(4) Magazines
   a. Total magazines: 2
   b. Total scanned pages: 26

References

1. Amin, A.: Offline Arabic character recognition—a survey. Proc. IEEE, pp. 596–599 (1997)
2. Amor, N.B., Amara, N.B.: Multifont Arabic character recognition using Hough transform and hidden Markov models. In: Proceedings of the 4th International Symposium on Image and Signal Processing and Analysis (ISPA'05), Zagreb, Croatia, pp. 285–288
3. Arica, N., Yarman-Vural, F.T.: An overview of character recognition focused on off-line handwriting. IEEE Trans. on Systems, Man, and Cybernetics **31**(2), 216–233 (2001)
4. Basu, M., Bunke, H., Bimbo, A.: Guest Editors' introduction to the special section on syntactic and structural pattern recognition. IEEE-TPAMI **27**(7), 1009–1012 (2005)
5. Bender, M.: Language in Ethiopia. Oxford University Press, London (1976)
6. Bigun, J.: Vision with Direction: A Systematic Introduction to Image Processing and Vision. Springer, Heidelberg (2006)
7. Bigun, J., Bigun, T., Nilsson, K.: Recognition by symmetry derivatives and the generalized structure tensor. IEEE-TPAMI **26**(2), 1590–1605 (2004)
8. Bigun, J., Granlund, G.H.: Optimal orientation detection of linear symmetry. In: Proceedings of IEEE First International Conference on Computer Vision. London, pp. 433–438 (1987)
9. Bunke, H.: Hybrid pattern recognition methods. In: Bunke, H., Sanfeliu, A. (eds.) Syntactic and Structural Pattern Recognition: Theory and Applications. World Scientific, Singapore (1990)
10. Bunke, H.: Recognition of cursive roman handwriting—past, present and future. In: Proceedings of the 7th International Conference on Document Analysis and Recognition, vol. 1, Edinburgh, 2003, pp. 448–459
11. Bunke, H.: String matching for structural pattern recognition. In: Bunke, H., Sanfeliu, A. (eds.) Syntactic and Structural Pattern Recognition: Theory and Applications. World Scientific, Singapore, 1990
12. Chen, C.H., DeCurtines, J.L.: Word Recognition in a Segmentation-free Approach to OCR. Proc. IEEE, pp. 573–576 (1993)
13. Cowell, J., Hussain, F.: Amharic character recognition using a fast signature based algorithm. In: Proceedings of Fourth International Conference on Information Visualization, pp. 384–389 (2003)
14. Dreyfus, G.: Neural Networks: Methodology and Applications. Springer, Heidelberg (2005)
15. Dutta, A., Chaudhury, S.: Bengali alpha-numeric character recognition using curvature features. Pattern Recognit. **26**(12), 1757–1770 (1993)
16. Fujisawa, H., Nakano, Y., Kurino, K.: Segmentation methods for character recognition: from segmentation to document structure analysis. Proc. IEEE **80**(7), 1079–1092 (1992)
17. Gatos, B., Karras, D., Perantonis, S.: Optical character recognition using novel feature extraction and neural network classification techniques. Proc. IEEE, pp. 65–72 (1994)
18. Gerard, A.: African Language Literatures: An Introduction to the Literary History of sub-Saharan Africa. Three Continents Press, Washington (1981)
19. Ha, T., Bunke, H.: Image Processing Methods for Document image Analysis, Handbook of Character Recognition and Document Image Analysis. World Scientific, New Jersey (1997)
20. Hagedoorn, M.: Pattern matching using similarity measures. PhD Thesis, Utrecht University, Utrecht, Netherlands (2000)
21. Ho, T.K.: Multiple classifier combination: lessons and next steps. In: Bunke, H., Kandel, A. (eds.) Hybrid Methods in Pattern Recognition. World Scientific, Singapore (2002)
22. Jain, A., Duin, R., Mao, J.: Statistical pattern recognition: a review. IEEE-TPAMI **22**(1), 4–37 (2000)
23. Khorsheed, M.S.: Off-line Arabic character recognition—a review. Pattern Anal. Appl. **5**, 31–45 (2002)
24. Mori, S., Nishida, H., Yamada, H.: Optical Character Recognition. Wiley, New York (1999)
25. Mori, S., Suen, C., Yamamoto, K.: Historical review of OCR research and development. Proc. IEEE **80**(7), 1029–1058 (1992)
26. Navarro, G.: A guided tour to approximate string matching. ACM Comput. Surv. **33**(1), 31–88 (2001)
27. Premaratne, L., Assabie, Y., Bigun, J.: Recognition of modification-based scripts using direction tensors. ICVGIP'04, Kolkata, pp. 587–592 (2004)
28. Sanfeliu, A.: Matching tree structures. In: Bunke H., Sanfeliu A. (eds.) Syntactic and Structural Pattern Recognition: Theory and Applications. World Scientific, Singapore (1990)
29. Smith, P.: Applied Data Structures with C++. Jones & Bartlett, Sudbury (2004)
30. Singh, S., Amin, A.: Fuzzy recognition of Chinese characters. In: Proceedings of the Irish Machine Vision and Image Processing Conference (IMVIP'99), Dublin (1999)
31. Srihari, S.N., Hong, T., Srikantan, G.: Machine-printed Japanese document recognition. Pattern Recognit. **30**(8), 1301–1313 (1997)
32. Suen, C.Y., Mori, S., Kim, S.H., Leung, C.H.: Analysis and recognition of Asian scripts—the state of the art. In: Proceedings of the 7th International Conference on Document Analysis and Recognition, vol. 2, Edinburgh, pp. 866–878 (2003)
33. Trier, O.D., Jain, A.K., Taxt, T.: Feature extraction methods for character recognition—a survey. Pattern Recognit. **29**(4), 641–662 (1996)
34. Weickert, J.: Coherence-enhancing shock filters. In: Michaelis, B., Krell, G. (eds.) Lecture Notes in Computer Science, vol. 2781, Springer, Berlin, pp. 1–8 (2003)
35. Wu, X., Wu, M.: A recognition algorithm for chinese characters in diverse fonts. In: Proceedings of the 2002 International Conference on Image Processing (ICIP'02), vol. 3, New York, pp. 981–984 (2002)