The Use of Object-Oriented Approach for Arabic Documents Recognition

Ibrahim A. Albidewi
Computer Science Department, College of Computing and Information Technology
King AbdulAziz University
P.O.Box 9028, Jeddah 21413, Saudi Arabia

Summary
This paper presents a system for Arabic character recognition which is implemented using Object Oriented Programming (OOP). The system starts by scanning the document which will be processed to resolve the skewing problem, and then the document will be segmented into lines where each is segmented into words. Each word is segmented into characters or primitives also some characters will be fragmented during this process. The features of the fragments characters will be obtained and a neural network module will be used for the recognition. A finite state automata recognizer is used for recognizing the fragments of each character.

Key words:
Arabic Character Recognition, OCR, Objected-Oriented Programming, Neural Network, Document Segmentation, and Image Processing

1. Introduction

One of the common methods for converting written texts to electronic text is optical character recognition (OCR). A lot of work has been done on English OCR, but Arabic OCR is still under development [1] [2]. In almost every image processing application, preprocessing stage is required ranging from biometric analysis to document image analysis. An input image need to be normalized and converted into format acceptable by OCR system. OCR systems typically assume that documents were printed with a single direction of the text and that the acquisition process did not introduce a relevant skew [3] [4].

OOP principles and design patterns are introduced in many applications as the means to cope with design complexity [5] [6]. Many articles introduce a software development framework, which amasses object-oriented programming (OOP) concepts and designed procedures, intended to systematize the implementation of link-level tools. This development framework is fully implemented in C++ programming language providing modularity and reusability (improving the coding activity). This framework then constitutes remarkable tool for quickly creating link-level applications [7].

Manuscript received April 5, 2008
Manuscript revised April 20, 2008

The ultimate goal for character recognition in general is to develop a communication interface between the computer and its users. This implies the direct storing of handwriting of the users into computer memory without going through a keyboard and enable the users of computer systems to store Arabic documents directly to memory as text file. These text files can be accessed and edited later for more processing if desired. The scope of Arabic Character Recognition varies from the simplest form of isolated printed characters to the most complex one of the hand-written documents. Nowadays, researchers strive toward achieving more speed for recognition with higher accuracy [8]. This becomes available by means of sophisticated and much more powerful PC's. Although Arabic alphabet contains only 28 characters, yet the process of recognition deals with more than 60 characters. This is because Arabic characters take different shapes depending on their location within the word.

Table (1) [9] indicates the different shapes of each character when it is located at the start, middle, or end of the word. The character also has a different shape if it is isolated. Also, some characters are classified into some groups having the same main stroke with minor change. This change is indicated by having different number of dots as well as their location with respect to the main stroke.

Arabic handwritten characters are written cursively. Therefore, segmentation of Arabic handwritten words is associated with some problems. These are presented in overlapping of sub words, different sizes of written characters, and the existence of connecting strokes with different length between the written characters.

In this paper, the block diagram of the system consists of four parts as shown in Figure 1. The first part is called the page analyzer. The seconded one deals with functions that extract the features. The third deals with four neural networks to recognize the characters or fragments while the last deals with a finite state automata (FSA) recognizer. Moreover, figure 1 gives the block diagram of the system

while figure (2) shows the relationships among those modules.

2. Page Analysis Module

This module receives the scanned document pages which is accomplished by means of a TWAIN package used for the scanner. Therefore, the scanner can be operated from within the Module to produce the bitmaps of pages of a document.

The module performs the segmentation process of each Page into Lines, Words and Characters. It also builds objects of the class CPageInfo that encapsulate the segmentation data and functions that accomplish the segmentation. These objects are exported in a persistent form for further processing by the FSA Recognizer Module.

In this work, a Page is defined to be a block of text lines using the same font. A Word is a sequence of connected Characters within the same Line. The segmentation process of a Word into Characters may cause the fragmentation of some standard characters into several fragments of a character. Therefore, the set of all possible standard characters as well as all possible fragments define our Character set.

2.1 Representation of Segmentation

The segmentation hierarchy is represented by maintaining a linked list of CLineInfo objects within CPagelnfo. CLinelnfo in turn maintains a linked list of CWordlnfo objects. The CLineInfo objects keep the data for allocating the bounding rectangle of each Line within the Page bitmap as well the position of the base line zone for the line of text. The CWordlnfo objects keep similar data for the Words within the Line as well as the x-positions at which each word is segmented into Characters. The CWordlnfo objects also store the individual Character codes (ISO codes / UNICODE values) which will be filled by the Recognizer Module.
Object Persistence is implemented by defining the member function Serialize for each of the CPageInfo, CLineInfo and CWordlnfo classes the linked list objects are taken from the MFC (Microsoft Foundation Classes) Collection class CObList that has a predefined Serialize member function. This design benefits from MFC code reusability and allows for exporting the objects in a persistent form for further processing by the FSA Recognizer Module.

2.2 Segmentation Process

This process mainly consists of three types of operations. The first operation deals with a page to line segmentation. The second one separates a line into its words and the third one segments a words into characters or parts of a character.

The Page to Lines segmentation process is implemented by an object of the class CPage Lines Analyser. This class encapsulates a pointer to the CPagelnfo object and the working data used by the member functions that perform the segmentation. The segmentation process uses horizontal projection of the bitmap pixels to form an X-Histogram. The histogram zones define the Lines zones. Moreover, the maxima within the histogram zones define the Base Line of each line within the text.

The Line to Words segmentation process is implemented by an object of the class CLineWordsAnalyser. Therefore, this class encapsulates a pointer to the CLineInfo object and the working data used by the member functions that perform the segmentation. The segmentation process uses the vertical projection of the bitmap pixels to form a Y-Histogram. The histogram zones define the Words zones.

The Word to Characters segmentation process is implemented by an object of the class CWordCharsAnalyser. This class encapsulates a pointer to the CWordInfo object and the working data used by the member functions that perform the segmentation. The segmentation process uses a vertical projection of the bitmap pixels while masking the base line zone to form Masked-Base-Line (Y–Histogram). The histogram zones define the individual Characters or the Character fragments.

2.3 Page Deskewing

The Masked-Base-Line Word to Characters segmentation process is sensitive to the correct determination of the base line zone. A very useful feature of Arabic printed text is the presence of pronounced peeks of the X-Histogram at the locations of the base line. However, a correction must be done for any skew in the Page bitmap. The deskewing operation should be implemented in order to rely on this feature for correct Word to Characters segmentation.

The Page Analysis Module uses an object of the class CPageDeskewer to rotate the bitmap after estimating the required angle of the rotation. The member function GetLinelnclillationO determines the average inclination of the Page base lines. The average inclination is determined by using a modified version of the Hough transform, where the points are transformed in the 'xy-plane to lines

in the mc-plane (i.e. the point (Xi, Yi) lies on a line (M Xi + C = Yi) which represents a line Plane.

The member function GetWordBaseLinesO is used to build the X-Histogram and determine the base line of each Word. This is followed by transforming the midpoints of each Word's base line into a line in the mc-plane. GetLinelnclination() then uses the least squares method to determine the best m value that fits the given data. This m value is the average of the actual base lines inclinations to be corrected for. The above produces the angle of the rotation to be used by the member function DeskewBitmap() and the last function performs the actual rotation in a pixel-wise fashion to bring the base lines to the determined horizontal positions.

3. Features Extractor Module

As mentioned before, each character of the characters set is represented by a bitmap file. Moreover, this set is prepared in a common disk directory, such that for each Character the following data is also maintained**:**

Character Identifier,
Code (ISOIUNICODE value),
Bitmap file name,
*Character Context**.***

The following shows some sample Character Specifications,

ط (Tah)	Height = 41, Width = 16, Code = 216, Context = BMSE
ع (Ain_E)	Height = 48, Width = 16, Code = 218, Context = E
غ (Ghain_M)	Height = 42, Width = 10, Code = 219, Context = M

where character context is used to specify the possible location of the Character with respect to the Word where it belongs to. This can be as follows**:**

a) Beginning of the Word denoted by B,

b) End of the Word denoted by E,

c) Middle of the Word denoted by M,

*d) Separate position of the Word denoted by S**.***

The Features Extractor Module stores this data for each *Character* in the character set in *CCharInfo* objects. Those objects are stored in a linked list within a *CFontInfo* object. The Module will then build a *CCharAttributes* object that encapsulates the features it extracts for each Character. Again, the *CCharAttributes* objects for the Character set are stored in a linked list within a *CFontAttributes* object**.**

The principle of information hiding in Object-Oriented design model allows accommodating a variety set of features. This enables the system to use a different design for the data structure within the *CCharAttributes* objects depending on the set of features used in the recognition process. All it needs is that each design must implement the following member functions**:**

GetBPNlnput
*This function submits the Character features representation to be used as input to the Neural Network**.***

GetCharCode
*This function returns the value of Character Code desired as corresponding olitput from the Neural Network**.***

Serialize
This function implements object persistence so that the CFontAttributes object together with its component CCharAttributes objects can be exported to the Neural Network Generator and the FSA Recognizer Modules.

It is also specify that *CFontAttributes* object implements the member function**:**

BuildCharAttr
Given the bounding rectangle of a Character within a Bitmap object, this function builds a corresponding CCharAttributes object.

4. Neural Network Generator Module

This Module is responsible for building a *CBPNetwork* object that encapsulates a Back-Propagation Neural (BPN) Network. It is actually used to build four BPN Networks (for the Characters with B, M, E and S Context). The results of these networks are then exported to be used by the FSA Recognizer Module**.**

The Neural Network structure is done such that the user enters its specifications. Therefore, the number of hidden

layers, the numbers of Nodes and the type of the output function in each layer are determined by the user. The user can also specify if the Nodes use bias terms. Operational parameters such as the learning rate must also be specified.

Four BPN Networks (for the Characters with B, M, E and S Context). The results of these networks are then exported to be used by the FSA Recognizer Module. The Neural Network structure is done such that the user enters its specifications. Therefore, the number of hidden layers, the numbers of Nodes and the type of the output function in each layer are determined by the user.

The user can also specify if the Nodes use bias terms. Operational parameters such as the learning rate must also be specified.

After selecting the training *Characters* set, the user can start the training process while monitoring its convergence. The user can interrupt the training process to change the learning rate, or to add more nodes to the hidden layers, or simply save the Network in a persistent form to continue training at a later session.

The input to the Network represents the feature parameters encapsulated in a *CCharAttributes* object of some *Character*, either standard or to be recognized. The output is the bits representation for the *Character* code.

The *CBPNetwork* object maintains a linked list of *CBPNLayer* objects. Each of these *CBPNLayer* objects maintains an array of objects. The *CBPNetwork* object also maintains an array of *CTrainingExemplar* objects each of which encapsulates a pair of input and output vectors and the corresponding *Character* code. The input vector is a representation of the *Character* Attributes as determined by the Features Extractor Module. For this purpose, the Module imports the *CFontAttributes* object exported by the latter. The main member functions of *CBPNetwork* are:

AddTrainingExemplar
Called when defining the Character set to be used for training the Network.

DoTrainingCycle
Called to perform the Network training.

GetBPNOutput
Called when recognizing a Character within the scanned document.

Serialize
Called when exporting the Network to the FSA Recognizer Module.

The main member functions of *CBPNLayer* are:

SetInputs
Sets the inputs at the Nodes of the input layer.

ComputeOutputs
Computes the outputs at the Nodes of the layer.

ComputeErrors
Computes the errors at the Nodes of the output layer.

BackPropagateErrors
Computes the back-propagated errors.

UpdateWeights
Updates the weights of the connections of the Nodes in this layer to the Nodes of its previous layer.

AddNewNode
This function allows for adding new Nodes to the layer during training as dictated by convergence behavior.

The last function allows for reaching a near optimal number of Nodes in the hidden layers, while starting from a small number and allowing the Network to grow as necessary. The main member functions of *CBPNNode* are:

ComputeOutput
Called by the implementation of CBPNLayer: :ComputeOutputs.

BackPropagateError
Called by the implementation of CBPNLayer: :BackPropagateErrors.

UpdateWeights
Called by the implementation of CBPNLayer::UpdateWeights.

AllocateWeights
Used for first-time allocation of memory for the weights associated with this Node.

ReAllocateWeights
Used to allocate additional memory as required when new Nodes are added.

5. The FSA Recognizer Module

The main objective of this module is to compose the fragments of a character produced by segmentation into a complete one.

6. Conclusion

A system for recognizing Arabic text is given. It is adopted the concepts of object oriented programming and implemented using Visual C++. The principle of information hiding in Object-Oriented design allows supporting a variety set of features. This enables the system to use a different design for the data structure within the object specified for this purpose as long as a set of member functions are implemented. For the future consideration, the proposed system should have the ability to deal with documents that include some figures and tables.

Acknowledgments

I would like to thank my colleagues Dr. K. Jambi and Prof. R. Amer for their advice and support throughout. This work was supported by King Abdulaziz City for Science and Technology (KACST - project number AR-15-20).

References

[1] Shirali-Shahreza, M.H., Shirali-Shahreza, S., "Persian/Arabic Text Font Estimation using Dots", IEEE International Symposium on Signal Processing and Information Technology 2006 , pp. 420–425, Aug. 2006.

[2] Al-Shoshan, A.I., "Arabic OCR Based on Image Invariants", Geometric Modeling and Imaging--New Trends, 2006, pp. 150-154 , July 2006.

[3] Sarfraz, M., Shahab, S.A., "An efficient scheme for tilt correction in Arabic OCR system", International Conference on Computer Graphics, Imaging and Vision: New Trends, 2005, pp. 379–384, July 2005.

[4] Zidouri, A, "ORAN: a basis for an Arabic OCR system", International Symposium on Intelligent Multimedia, Video and Speech 2004, pp. 703-706, Oct 2004.

[5] Binkley, D., Ceccato, M., Harman, M., Ricca, F., Tonella, P., Loyola Coll., Baltimore, MD, "Tool-Supported Refactoring of Existing Object-Oriented Code into Aspects", IEEE Transactions on Software Eng., Volume 32 , Issue 9, pp. 698 –717, Sept 2006.

[6] Woei-Kae Chen, Yu Chin Cheng, "Teaching Object-Oriented Programming Laboratory With Computer Game Programming", IEEE Transactions on Education, Volume 50, Issue 3, pp. 197–203, Aug 2007.

[7] Debuse, Justin C. W., Stiller, Tony, "Technologies and Strategies for Integrating Object-Oriented Analysis and Design Education with Programming", Australian Conference on Software Engineering, ASWEC 2008 19th, pp. 97–103, March 2008.

[8] Miletzki, U., "Character recognition in practice today and tomorrow", Proceedings of the Fourth International Conference on Document Analysis and Recognition, 1997, vol.2, pp.902-907, Aug. 1997.

[9] Jambi, Kamal, "Recognition of Constrained Handwritten Arabic Words Without Segmentation", Proceedings of the 6th International Conference on Computer Theory and Applications, Alexandria, Sept. 1996.