TECHNISCHE UNIVERSITÄT CAROLO-WILHELMINA ZU BRAUNSCHWEIG

Bachelorarbeit

Effect of System Parameters on Feature Extraction Sets for Arabic Handwritten Text Recognition

Asier Diez Iztueta

9. September 2011

Institut für Nachrichtentechnik
Prof. Dr.-Ing. Tim Fingscheidt

betreut durch:
Akad. Direktor Dr.-Ing. Volker Märgner
Dipl.-Inform. Haikal El Abed

Abstract

The purpose of this study is to analyze the effect of different parameters on a Pattern Recognition System for Arabic Handwritten Text Recognition, and perform different experimental tests in order to obtain the optimum values for the process.

In the first introductory section, source data material and the tools used in this work are introduced and explained.

The thesis then focuses on the Feature Extraction Process, providing details about different strategies or methods that can be used on the process. In the experimental section, the most important test results are given and the variable parameters are individually analyzed. Finally, different combination schemes are implemented in order to prove the effectiveness of the Slanted Windows.

The results provide some support for the correct selection of parameter values for future implementations of the system. However, the optimum parameter values should not be considered as absolute values, due to the fact that the aim is to guide the researchers for future implementations of the system.

Contents

List of Tables

List of Figures

1 Introduction

In this introductory chapter, the main subject and goals of this bachelor's thesis are presented. In addition, this section will provide a general overview of the field treated.

1.1 Motivation

This bachelor's thesis consists on an experimentation with the Feature Extraction section of a Pattern Recognition System. The work is be based on the feature set designed by Ramy Al-Hajj Mohamad, Laurence Likforman-Sulem and Chafic Mokbel [1], and considers the following points as the main goals:

- Analyze the effect that different input parameters can have in a Pattern Recognition System, particularly in an Arabic Handwriting Recognition System like the one introduced in the paper [1]. Observe how these parameters affect on the feature extraction process and on the final recognition rate.
- Propose alternative methods for implementing the Feature Extraction Process.
- Experiment with the Slanted Window explained in paper [1] and observe the improvements that it provides to the System.
- Analyze the effectiveness of different combination methods which merge the results given by different sliding windows.
- Construct solid conclusions drawn from the tests and experiments for further implementations of the Feature Extraction Process.

1.2 Pattern Recognition System

Having as input some sort of data, Pattern Recognition Systems are responsible for obtaining some specific values or labels as a result. There are many kinds of algorithms for constructing a Pattern Recognition System, and particularly, the Statistical Classification Algorithm is the one which is used in the system presented in this thesis. These classification algorithms classify each input value or label into one of a given set of classes [13].

In the Figure 1.1, a general diagram for a Pattern Recognition System is shown. There are many fields where these systems can be implemented on. For example, medical imaging, speech recognition, or handwriting recognition, being the latter the one which is treated in this thesis. Normally, a Pattern Recognition System consists on the following sections:

1. Acquisition.
2. Data.
3. Pre-processing.
4. Segmentation, feature extraction.
5. Classification, recognition.
6. Post-processing.

This work concerns the Feature Extraction section. It is worth to say that Feature Extraction is not a fixed process, and the choice of suitable features can be different depending on the objects that have to be described. For instance, a proper recognition system for Latin Handwriting should use a different feature set from a recognition system for Chinese Handwriting, due to the different properties that each kind of handwriting has. Therefore, the selection of each feature set should be optimized based on the properties of the objects that the system has to describe [1].

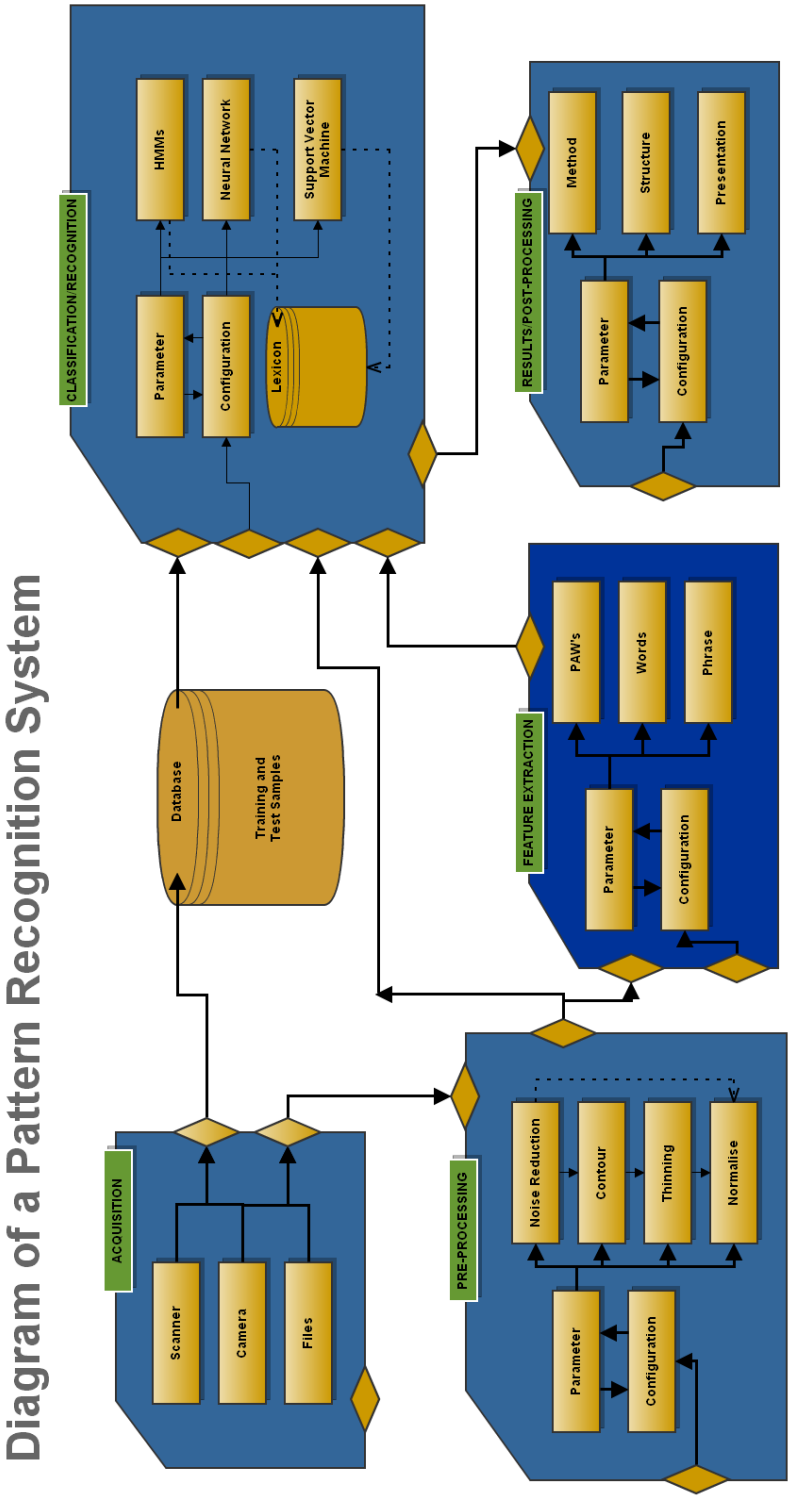


Figure 1.1: *Diagram of a Pattern Recognition System.*

1.3 MATLAB

The programming tool used in this bachelor's thesis is MATLAB 7.11.

MATLAB is a numerical computing environment and a fourth-generation programming language. The software allows matrix manipulations, plotting of functions or data and implementation of algorithms, amongst other versatile processes [14]. Matrix manipulation implies image manipulation, as images are stored in the form of digital matrixes. Morover, there are additional toolboxes which can be used, such us the Image Processing Toolbox, which allows working with images in a sofisticated way. For instance, this toolbox offers the following features [15]:

- Image enhancement, filtering, and blurring.
- Segmentation, morphology, feature extraction, and measurement.
- Spatial transformations and image registration.
- Image transforms (FFT, DCT, Radon) and fan-beam projection.
- Multidimensional image processing

Taking into account all these features, MATLAB was chosen as the most suitable programming software for implementing and analyzing the Feature Extraction step.

1.4 Arabic Handwriting

The Object: Tunisian City Names The main goal of the Recognition System presented in [1] is to recognize handwritten Arabic city names.

For the training and testing process, the IFN/ENIT database was used. This database consists of 946 Tunisian town/village names written by 411 different writers. In total, there are about 26.400 names which contain more than 210.000 characters in overall [7]. The given images are binary and saved in the TIF format, after being pre-processed. An example of how these images look like is given in the Figure 1.2.

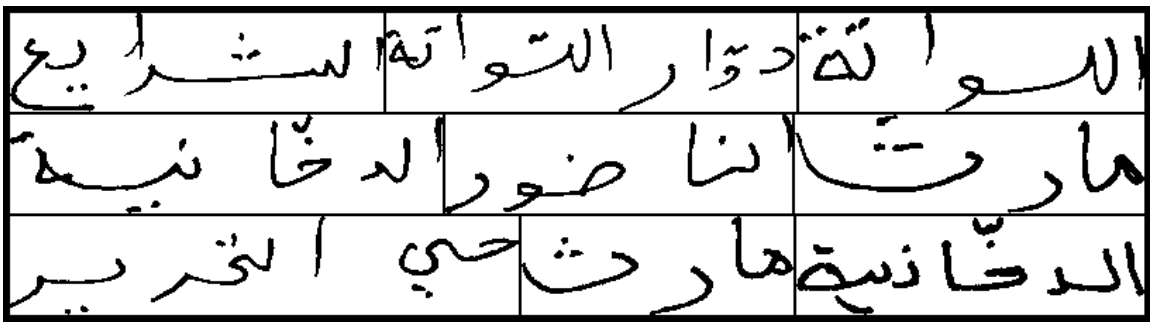


Figure 1.2: *Example of 9 handwritten images from the IFN/ENIT Database.*

Characteristic of the Arabic Handwritting What it is special about the Arabic alphabet is that although it only contains 28 characters, the number of shapes is much higher, being exactly 109 [3]. The reason of having this high amount of shapes is that each character is written differently depending on the position it has in the word. Thus, we will have different shapes for the same character depending on if it is located in the beginning of the word, in the middle, in the end or it an isolated character [1]. All these shapes are shown in the Figure 1.3.

There are also additional characteristics such as dots and diacritical marks that change the letter and the word meaning, and other marks which are used to indicate vowels. Also, additional shapes are created by ligatures [1].

One of the most distinctive features of the Arabic Handwriting is the horizontal baselines. The lower baseline is the line where ligature occurs between characters, being usually the densest line in word images. The characteristic descending and ascending traces extend from these two baselines [5].

Baselines are going to be analyzed further in chapter 2.4 on page 10.

Name	End	Middle	Beginning	Isolated
ʾalif	ـا	ـا	ا	ا
bāʾ	ـب	ـبـ	بـ	ب
tāʾ	ـت	ـتـ	تـ	ت
ṯāʾ	ـث	ـثـ	ثـ	ث
ǧīm	ـج	ـجـ	جـ	ج
ḥāʾ	ـح	ـحـ	حـ	ح
ḫāʾ	ـخ	ـخـ	خـ	خ
dāl	ـد	ـد	د	د
ḏāl	ـذ	ـذ	ذ	ذ
rāʾ	ـر	ـر	ر	ر
zayn / zāy	ـز	ـز	ز	ز
sīn	ـس	ـسـ	سـ	س
šīn	ـش	ـشـ	شـ	ش
ṣād	ـص	ـصـ	صـ	ص
ḍād	ـض	ـضـ	ضـ	ض
ṭāʾ	ـط	ـطـ	طـ	ط
ẓāʾ	ـظ	ـظـ	ظـ	ظ
ʿayn	ـع	ـعـ	عـ	ع
ġayn	ـغ	ـغـ	غـ	غ
fāʾ	ـف	ـفـ	فـ	ف
qāf	ـق	ـقـ	قـ	ق
kāf	ـك	ـكـ	كـ	ك
lām	ـل	ـلـ	لـ	ل
mīm	ـم	ـمـ	مـ	م
nūn	ـن	ـنـ	نـ	ن
hāʾ	ـه	ـهـ	هـ	ه
wāw	ـو	ـو	و	و
yāʾ	ـي	ـيـ	يـ	ي

Figure 1.3: *Arabic letters and shapes.*

2 Feature Extraction Process

The content in this chapter analyzes different strategies and methods for the Feature Extraction Process. Then, a specific Feature Set is explained in detail and different ways of computing it are discussed.

2.1 Feature Extraction

Feature Extraction is a process which allows an object to be recognized without having to take into account the whole information that the object provides. When humans try to recognize an object, the brain rejects all the information which is unnecessary, and selects only certain information which is the key to recognize the object. This information can be the color, the shape, the size, the texture... and the fact that these properties are the same for two different objects which belong to the same classification group, makes these characteristics to be *features*.

In a digital Pattern Recognition System, features must be created computationally to construct a certain Feature Set. The main problem in the design of a Feature Set is the appropriate selection of particular features. This selection must be very carefully made, as it is a very important task to think about which features could be the best and the most efficient. To achieve this goal, the following points have to be taken into account:

1. Previous to the computational design of the Feature Set, one must think of features which would be the best for distinguishing between different classes.

2. Features must be computationally implemented in an efficient and a proper way.

3. The number of features must be low enough in order not to demand a high computational power or a large amount of memory to the system. In contrary, the number of features must be high enough to be able to separate between different classes [2].

4. Feature values must be similar for words in the same class and different for

words belonging different classes.

2.2 Analytical and Holistic Strategies

Basically, techniques for handwritten word recognition can be split up in two general groups: *analytical strategy* and *holistic strategy* [1].

In the *analytical strategy*, a segmentation process is carried out previous to the feature extraction process. This segmentation process extracts individual letters for character recognition. In the *holistic strategy*, however, features are extracted from word images directly, these not being segmented into smaller units. For Arabic Handwriting Recognition, the *holistic strategy* is recommended above the analytical strategy, due to the fact that ligatures make it difficult the task of segmenting whole words into characters [1].

Disadvantages for *holistic strategy* are on one hand that vocabulary is limited for giving exact results, and on the other hand, that a handwritten word can be written in many different ways, this meaning that the range of possibilities for writing the same word is much wider than the range of possibilities for writing a single character.

On this bachelor's thesis, the parameters used in the feature set proposed by Ramy Al-Hajj Mohamad, Laurence Likfoirman-Sulem and Chafic Mokbel is going to be analyzed, which is based in the *holistic strategy*. The system proposed relies on the combination of a reference HMM-based classifier with two other classifiers which are the modified versions of the reference classifier.

2.3 A System for Feature Extraction and Handwritten Text Recognition

In this section, the mentioned Pattern Recognition System will be explained in detail. The general diagram for the system can be observed in the Figure 2.1.

As it is shown, the input for the system are preprocessed binary images. This images go through these following steps in the system:

1. Baseline Estimation.

2. Feature extraction.

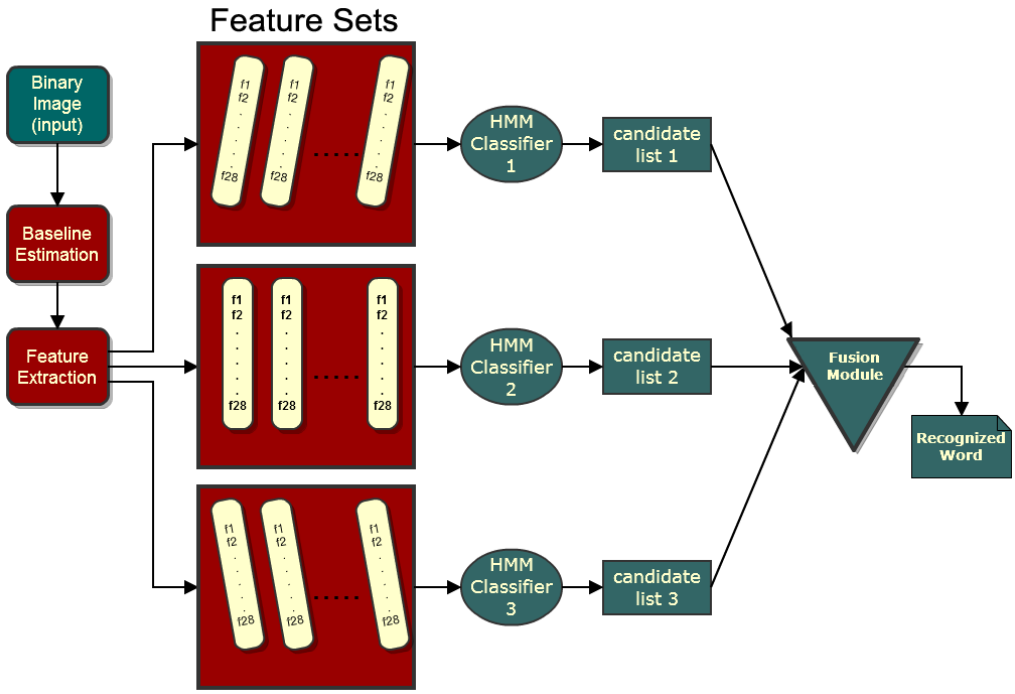


Figure 2.1: *Diagram of the Pattern Recognition System on paper [1].*

3. HMM (Hidden Markov Models).

4. Fusion Module.

First, baselines are estimated in each image. The main function of the baselines is to divide the image in 3 different zones: the upper zone, the core zone and the lower zone. 11 out of 28 features (39,29%) will extract information of the handwriting using this partition of the image.

After the horizontal baselines are extracted, a sliding window will be operated upon the images and the 28 features will be calculated from them. The width of the sliding window determines the number of frames in which the images will be split up into, building up a 28-feature vector per frame. This sliding window operates with 3 different inclinations, in order to adapt to different stroke directions.

In the end, different results obtained by different sliding windows will be combined and introduced to the HMM classifiers, which will give an answer for each of the images and a general recognition rate. Hidden Markov Models (HMMs) were chosen because of its ability to cope with variable length observation sequences and with nonlinear distortions. This ability makes HMM to be suitable for cursive handwriting such as Arabic, Latin, and Korean scripts [8] [10] [4] [11] [12].

2.4 Baseline Extraction

As mentioned in the section 1.4, the horizontal baselines are a distinctive feature for the Arabic handwriting. What makes the extraction of these lines interesting is that they separate vertical ascending strokes from the ones which are descending, and also these two zones from the core zone. This way, these three distinctive parts could be individually treated in the Feature Extraction Process.

The system which is used in the paper [1] for extracting both baselines, is based on the algorithm described in [6]. Basically, vertical projection profile is calculated by summing pixel values along the horizontal axis. The position of the peak of the projection profile will determine the location of the lower baseline. Thus, the lower baseline will be the image line which is most dense in the whole image.

For determining the upper baseline, the image is scanned beginning from the top of the image, line by line. The density of each line is counted, and when this density contains a value which is greater than the average line density, then it will be defined as the upper baseline. An example for the upper and lower baselines is given in the Figure 2.2.

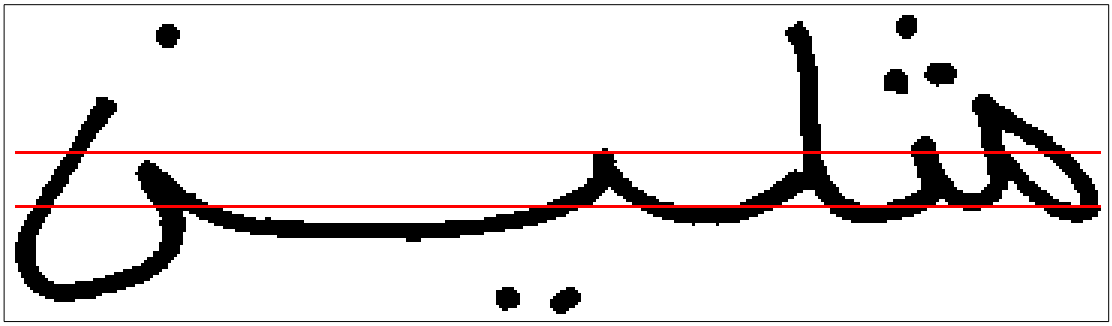
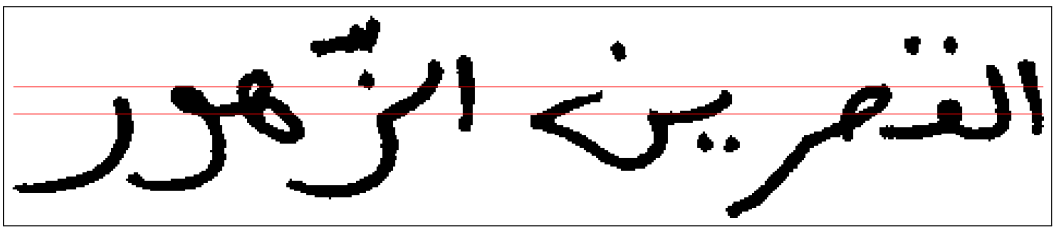


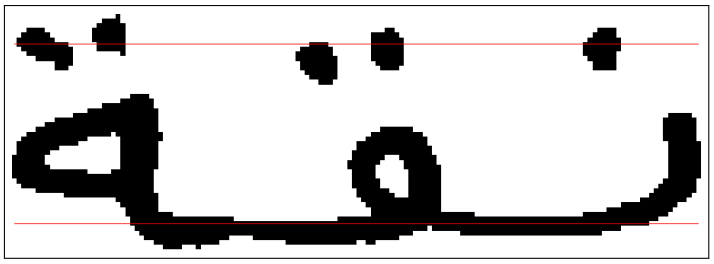
Figure 2.2: *Example of baseline extraction. Upper and lower baselines appear in red.*

Baseline Extraction Problems The result of the Baseline Extraction will not always lead us to the desired result. The reasons for a bad extraction of the baselines could be the following:

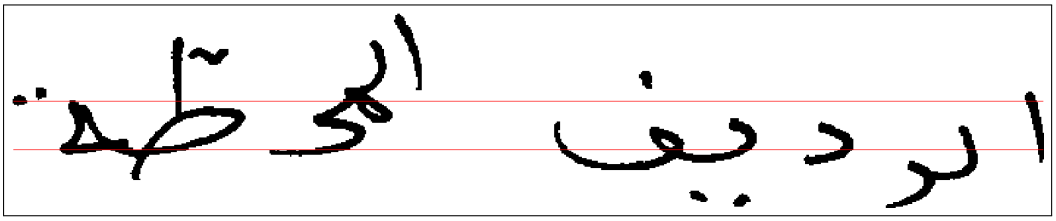
1. For extracting the upper baseline, the image is scanned from top to bottom. This does not happen with the lower baseline, as it is supposed that the densest line will be the lower baseline. Therefore, a wrong position for the baseline could be extracted. An example of this problem is illustrated in the Figure 2.3a.



(a) The densest line is not always the most representative.



(b) A high amount of diacritical points can bring a wrong baseline extraction.



(c) Baselines are not always representative in the whole image.

Figure 2.3: *Problems in the Baseline Extraction.*

2. A high amount of diacritical points can result on a non-desired upper Baseline Extraction. Figure 2.3b shows an example.

3. Sometimes the handwritten text is not perfectly aligned with the horizontal axis. The ideal baselines for this kind of images should be slanted too. However, the algorithm does not attempt to extract non-horizontal baselines. Figure 2.3c illustrates this problem, as the lower baseline does not effitiently represent the second half of the image.

Solutions for the Baseline Extraction Problems In order to extract better representative baselines, two improvements are proposed to the original algorithm:

1. For extracting the lower baseline, the image could be scanned from bottom to top, like the process for extracting the upper baseline. When a line is found with a density higher than the 80% respect the maximum density, this will be considered as the lower baseline. Instead of 80%, different values can also be used. This improvement can be seen in Figure 2.4a.

2. Normally, the upper baseline extracted is higher than the expected result, this caused many times because of the diacritical points. Therefore, it would be convenient the upper baseline to be related with a line situated lower in the image. In the original algorithm, the upper baseline is the first line in the top-down scanning which has a density which is bigger than the line with the average density. This problem could be partially solved if instead the average value, we considered a value which is 20% higher than the average density. The result of this improvement can be seen in *figures 2.4a* and *2.4b*. Again, 20% could not be the perfect value. Different values will be tested in the experimental section (2.4).

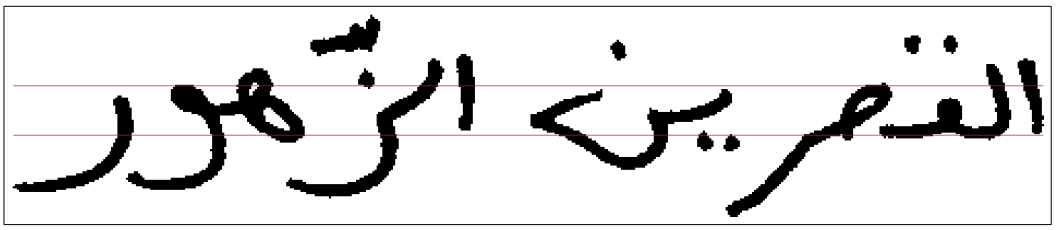
A new solution could be proposed to sort out the third problem. However, the handwritten words that the IFN/ENIT database contains are normally horizontal, and the average orientation is about 1,36° [1]. Therefore, only few images would be affected by this problem.

Test Results The main function of the baselines is to divide the image in 3 different zones, as mentioned above. This way, some features can differ between different parts of the image, and extract the result treating each part in a certain way. Some other features, however, do not take this difference into account. The features which do use the Baseline Extraction for distinguishing between different parts are the following:

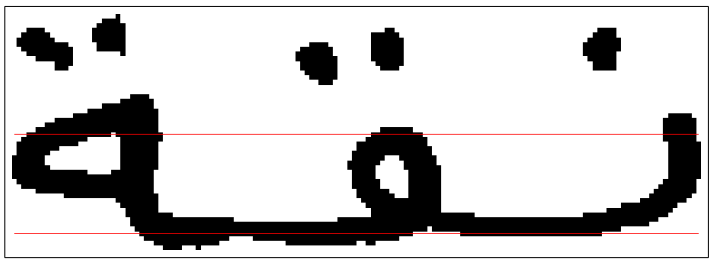
f12, *f13*, *f14*, *f15*, *f16*, *f23*, *f24*, *f25*, *f26*, *f27*, *f28* (39,29% of the features)

Due to that 39,29% is a high percentage, the process of the Baseline Extraction is considered as a very important step.

The proposed solutions (*section 2.4*) were implemented and tested in this work. The results for the solution n°1 is shown in Table 2.1, where the lower baseline is extracted with a bottom-top scanning. The percentage in the first row indicates the quantity respect the densest line which is going to be enough to consider a line as



(a) Baseline extracted with a bottom-top scanning.



(b) Implementing the second solution, diacritical points do not affect like before.

Figure 2.4: *Solutions for the Baseline Extraction Problems.*

the lower baseline.

Lower Baseline	100% (original)	75%	80%	85%	90%	95%
Result(%)	69.56	58.61	58.64	59.25	59.57	59.77

Table 2.1: *Recognition Rates with Bottom-Top baseline extraction.*

As it can be seen, the results with this new implementation are poor. Therefore, considering the densest line as the lower baseline like the original algorithm does, is a good criterion for extracting the lower baseline. This means that the problem n°1 can affect negatively only in a little amount of the images.

For the second proposed solution, the results obtained are shown in Table 2.2.

Lower Baseline	100% (original)	5%	10%	15%	20%
Result(%)	69.56	69.67	69.01	68.59	68.51

Table 2.2: *Recognition Rates with the Solution n°2.*

In these new experiments, it is demonstrated that the proposed solution for extracting the upper baseline can be helpful, as the result for the 5% percentage is slightly higher than the one achieved by the original algorithm. Therefore, this is a way with which the original algorithm could be improved. The percentages in the

first row indicate the values respect the average density which are summed in this new algorithm.

2.5 Feature Set

The feature set proposed by Ramy Al-Hajj Mohamad, Laurence Likforman-Sulem and Chafic Mokbel [1] is made up of 28 features which can be divided into two main groups: The first feature group contains 16 *distribution features*, and a big part of them provides information about the density of foreground pixels within the frame. The second feature group contains 12 *concavity features* which provide local concavity information and stroke direction within each frame. The percentage of each group respect the total number of features is shown in Table 2.3.

Distribution Features	Concavity Features
57,14%	42,86%

Table 2.3: *Amount percentages for each group of features.*

Distribution Features The descriptions for each feature corresponding to this group are given in Table 2.4.

Local Concavity Features This second group of features provide local concavity information and stroke direction within each frame. Thus, these features describe the concavities of the handwritten text and its handwriting orientation.

Local Concavity Features can be split up in two groups: Features 17 to 22 take the whole information in the image into account, while Features 23 to 28 consider the Core Zone only. The Core Zone is the part of the image between the Upper Baseline and the Lower Baseline.

In the Figure 2.5 six directions are shown. These are the directions which are going to be analyzed for the stroke orientation.

Feature 1	Feature 2	Feature 3	Features 4 to 11
Density of foreground pixels within the frame.	Number of black/white transitions between two consecutive frame cells.	Difference between the y-coordinate g of the center of gravity of foreground pixels of two consecutive frames t and t-1.	Densities of foreground pixels for each column in each frame.
Feature 12	**Features 13 and 14**	**Feature 15**	**Feature 16**
Vertical distance from the lower baseline of the center of gravity of foreground pixels, normalized by the height of the frame.	Density of foreground pixels over and (respectively, under) the lower baseline.	Number of transitions between two consecutive cells of different density levels above the lower baseline.	Zone to which the gravity center of foreground pixels belongs, with respect to the upper and lower baselines.

Table 2.4: *Descriptions for the Distribution Features*

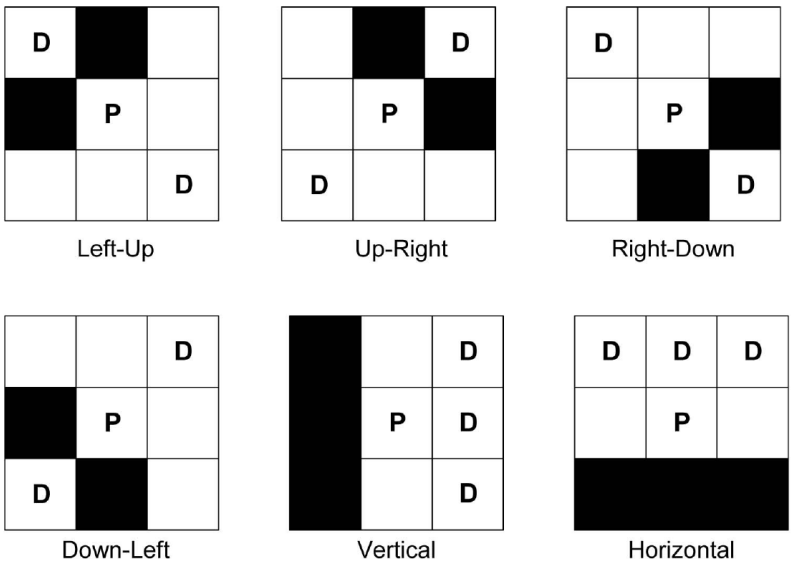


Figure 2.5: *Directions for Local Concavity Features.*

In this figure, P referes to the pixel which direction is being anylized, and D refers to the neighbour pixels which do not matter if they are black or white.

- **Features 17 and 23:** Normalized[1] number of hits in the direction Left-Up.
- **Features 18 and 24:** Normalized[1] number of hits in the direction Up-Right.
- **Features 19 and 25:** Normalized[1] number of hits in the direction Right-Down.
- **Features 20 and 26:** Normalized[1] number of hits in the direction Down-Left.
- **Features 21 and 27:** Normalized[1] number of hits in the Vertical direction.
- **Features 22 and 28:** Normalized[1] number of hits in the Horizontal direction.

2.6 Feature Extraction

When a sliding window operates in an image, 28 features are extracted from each frame, these beginning on the right side of the image and ending on the left of it, as shown in the Figure 2.7. The width of the window is fixed to 8 pixels (system parameter), while the height of it must be the same as the height of the image. These windows scan the image from right to left with a variable overlap parameter. The value of this parameter must have a value between 1 and k-1, being $k = 8$ pixels (fixed width of the window).

Apart from the baseline extraction which allows partitioning each image in three different parts, there is yet another division which is performed when each frame is extracted. Indeed, each frame is fragmented into nc number of cells, being nc another variable parameter of the system.

The features which are affected by these partitions are:

- **Affected by baselines:** *f12, f13, f14, f15, f16, f23, f24, f25, f26, f27, f28* (39,29% of the features).
- **Affected by cell fragmentation:** *f2* and *f15* (7,14% of the features).

Therefore, most of the features are not dependant from these two different fragmentations.

Another variable parameter of the system is *alpha* (α). This angle determines the slant angle of the sliding window. If *alpha* is positive, then the slanted window

[1] *Features 17* to *22* are normalized by the height of the image. *Features 23* to *28* are normalized by the height of the *core zone*

will be leaned to the right. In contrary, when *alpha* has a negative value, then the slanted window will be leaned to the left.

There are three sliding windows which are computed: one of them will be slanted with an angle of $+\alpha$, the second one will be implemented with a slant of the opposite angle ($-\alpha$), and the third Sliding Window will be vertical (α=0).

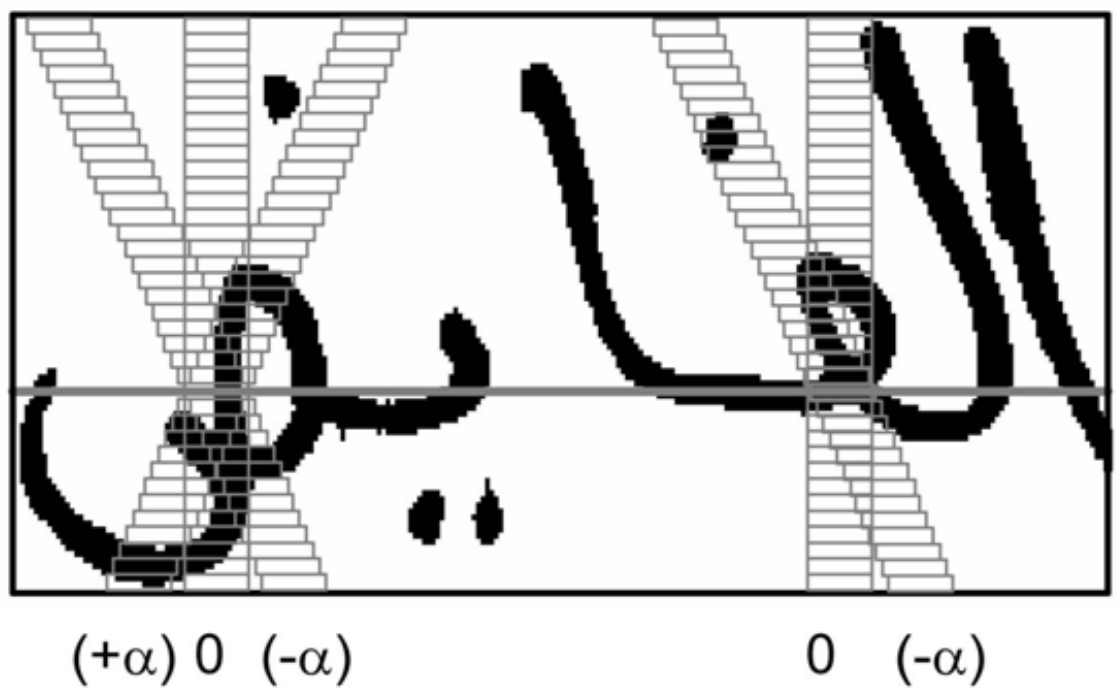


Figure 2.6: *Different orientations for the Sliding Window.*

Angle *alpha* is considered with respect to the vertical axis, as shown in the Figure 2.6.

In all cases, the first frame extracted will correspond to the very right side of the image, and the frames will be extracted consecutively towards the left, with a separation given by the offset parameter (see Figure 2.7).

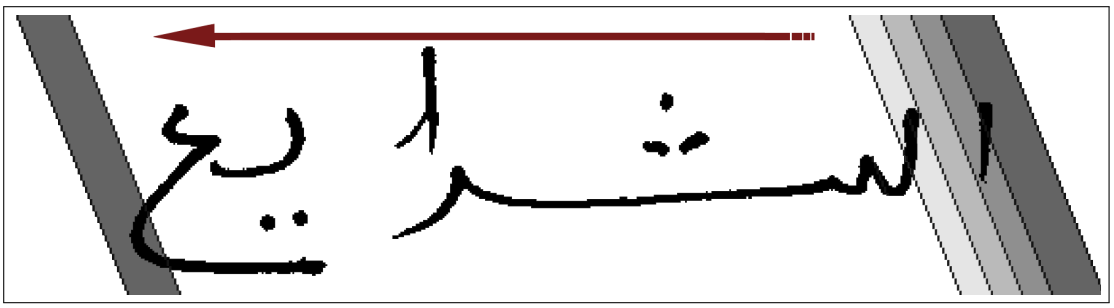


Figure 2.7: *Movement of the Sliding Window.*

2.7 Sliding Window

The role of the Slanted Windowing The concept of the Slanted Window was created with the goal of extracting characteristics in a way which better adapts to the orientation of the handwriting.

However, a problem exists when Slandted Windows are used: when the original image is not manipulated, not all the information in the image will be taken into account (see Figure 2.8). Therefore, using Slanted Windows without having changed the size of the original image can result in a bad recognition rate. This fact occurs because some information of the handwritten text is ignored. As the solution to this problem, background pixels are added on both sides of the image (Figure 2.9). The amount of the pixels added has to be enough to ensure that the whole handwritten text will be taken into account. This value corresponds to δ (delta).

$$\delta = H * tan[abs(\alpha)]$$

H: Height of the image in pixels.
α: Slant angle of the frames.

The word image of height H is thus enlarged by $H*tg(\alpha)/2$ on each side. With this process, now every pixel on the original image will be contained in a frame. The price to pay is that this way the image is bigger and therefore the computational process will be longer. Another inconvenient is that the added backround pixels do not contribute any useful information for the text recognition. However, as results have shown, the enlarged image results on much better recognition rates.

Test Results To evidence the fact explained above, two simple tests were carried out: in the first one, the size of the original images were not manipulated and there-

fore, the Sliding Windows were implemented on the original images (Figure 2.8). In this test, the result for the recognition rate achieved was 54,75%. In the second test, the image was enlarged in order to take into account the whole information of the handwriting (Figure 2.9). In this second test, the result achieved was 67,79%.

In conclusion, the test results demonstrate that is very recommendable to previously enlarge the images.

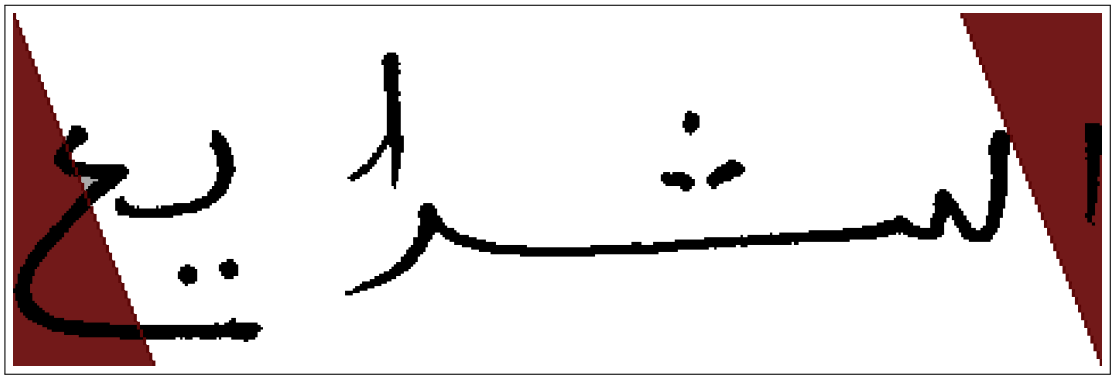


Figure 2.8: *Original image. Red zone represents the parts that the Sliding Window can not reach.*

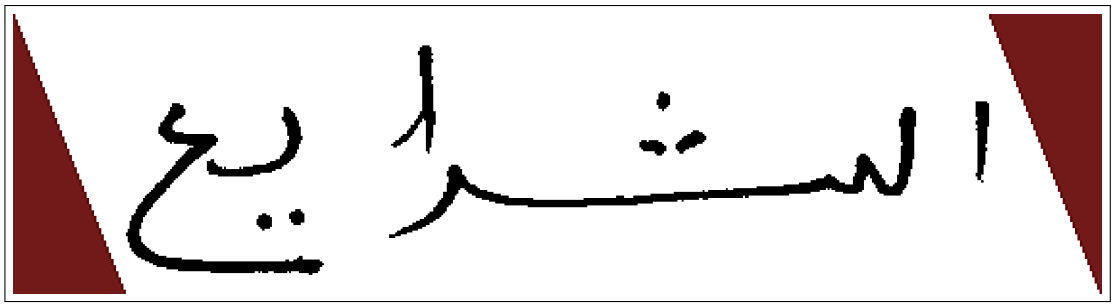


Figure 2.9: *Enlarged image. The Sliding Window can act in the whole image.*

2.8 Frame Extraction

Slanted frames have originally the shape of the type shown in Figure 2.10. As explained in section 1.3, MATLAB is a software which works with matrixes. Indeed, digital images are stored in the form of matrixes. When a frame is extracted, it has

to be saved in the form of a matrix and therefore, it will be stored in the form shown in Figure 2.11. This is done by concatenating each row in the frame. Note that this change in shape brings a slight change in the handwritten text.

A similar problem occurs when single frames have to be extracted.

This concatenation is not a problem for the *distribution features* (2.5). Their task is to extract information about densities and distribution properties, and they do not get affected by a change in the shape. However, the goal of the *local concavity features* (section 2.5) is to provide information about the stroke direction of the handwritten text. If the frame lines are concatenated, a problem exists because the shape of the handwritten text is not yet the same and therefore, the values for the *local concavity features* will be not reliable. Therefore, a new method has to be found to keep the original directions of the original handwritten text.

As it occurred in the previous chapter, the problem with the frames can be solved adding white pixels on the sides of the frame (*see Figure 2.12*). This way, the direction of the strokes will be still kept and be the same. In Figure 2.13, an example of another slanted frame is given, where the added pixels are shown in black in order to distinguish them from the original pixels in the frame.

The concatenation of the frame lines will not be treated as a problem as it has some good features too. For example, the computational speed is much quicker, due to it only has to compute a frame with the size equal to $w*H$, being w the fixed width of the frame (8 pixels) and H the height. On the other hand, the advantage of using the method to keep the original shape is that evidently, the original shape of the handwritten text is kept without any change and therefore, the features which provide information about the stroke direction of the handwritten text will be more reliable.

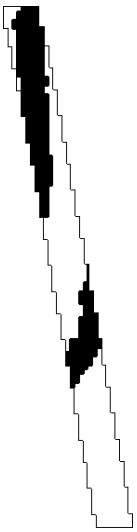


Figure 2.10: *Original Frame.*

Figure 2.11: *Concatenated Frame.*

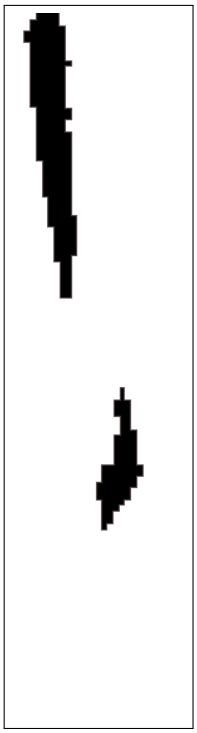


Figure 2.12: *White pixels added to the extracted frame.*

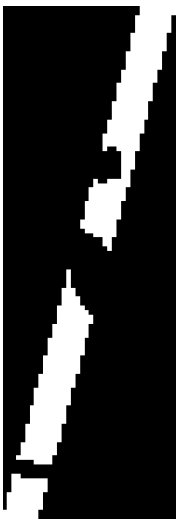


Figure 2.13: *Example of a Slanted Frame with a positive value for alpha.*

3 Experiments

In this chapter, different parameters for the Feature Extraction Process are analyzed by using different tests. Then, optimal values for the Feature Extraction will be given and different conclutions are going to be drawn out.

3.1 Experimental Setup

The first simulations and tests in this work were performed before having previously analyzed the parameters and their influence on the system. The references were simply taken from the authors of the paper [1], where, by their experiments, the following conclusions were drawn:

- The parameter number of cells (*nc)* is not very sensitive from values 17 to 30, being this range the most appropriate.
- The width of the sliding window is neither sensitive from values 6 to 12.

As the conclusion, the following values for the parameters were chosen as the best ones:

- **Frame width** = 8 pixels.
- **Offset** = 4 pixels.
- **Number of cells** = 21 pixels.

3.2 Variable Parameters of the System

So far, different ways for extracting frames and features have been analyzed. In detail, Baseline Extraction (section 2.4), Image Manipulation for Frame Extraction (section 2.7) and Frame Manipulation for Feature Extraction (section 2.8) have been reviewed. These mentioned chapters are related with the mechanisms of the system. However, there are other features which are called System Variable Parameters.

In this present section, the impact of the System Variable Parameters will be analyzed, in the sense of how much they affect on the system or how they can improve the recognition rate.

Being MATLAB the software where the feature extraction algorithm was written, there are unlimited possibilities for modifying the feature extraction algorithm, in order to achieve better results or to try different methods.

The variable system parameters will be split up into two main groups. On one hand, there are those parameters which were already defined in the original paper [1]. These are:

- **Number of cells:** Defines the number of cells in which each frame is built.
- **Offset:** This is the overlap parameter, defines the horizontal separation between two consecutive frames, in pixels.
- **Alpha:** Determines the slant angle of the Sliding Windows. When alpha is 0°, the Sliding Window will be vertical. If it is positive, the window will be leaned to the right and when it is negative, the windows are going to be leaned to the left.

On the other hand, there are different techniques and changes that were adopted in this work to make the system more versatile. These are:

- Image Height Normalization (*section 3.2.1*).
- Slant correction (*section 3.3*).
- UOB (*section 3.4*).

3.2.1 Image Height Normalization

The IFN/ENIT database consists of 24.000 images with different heights and widths. The fact that these sizes are not standarized can always lead to a bad recognition result, and therefore, it is always interesting to standardize the image sizes. Height Normalization was introduced as a technique that allows that all the images have the same height. The MATLAB programming software offers three different computation methods for scaling images and standarize the height:

METHOD NAME	DESCRIPTION
nearest	Nearest-neighbor interpolation. The output pixel is assigned the value of the pixel that the point falls within. No other pixels are considered.
bilinear	The output pixel value is a weighted average of pixels in the nearest 2-by-2neighborhood (bilinear interpolation).
bicubic	The output pixel value is a weighted average of pixels in the nearest 2-by-2neighborhood (bilinear interpolation).

Table 3.1: *MATLAB methods for rescaling the images.*

On the tests performed, the goal is to analyze the improvement that this Height Normalization technique can introduce, as well as comparing between different rescaling techniques given by MATLAB.

Test Results The recongnition rate obtained without having used the Height Normalization technique, was exaclty of 59,27

As said above, MATLAB offers 3 different approaches for rescaling the images and for deciding which one of them is the best. For this purpose, 9 tests were performed using 3 different values for the height normalization: 43 pixels, 65 pixels and 86 pixels. The recognition rates are shown in the Table 3.2.

- Recognition rate without using height normalization: 59,27%.

Comparing these results to the previous test in which no Height Normalization was used, it can be said that Height Normalization is one of the most important

Height	'nearest'	'bilinear'	'bicubic'
43 pixels	69,14%	68,86%	69,56%
65 pixels	64.25%	65.76%	65.51%
86 pixels	59.17%	59.45%	59.04%

Table 3.2: *Results for image scaling methods.*

improvements for the system, due to it can improve the result up to 10 points. In addition, it is an advantage that this technique can be computed with a single command line in MATLAB, what means that implementing the height normalization technique does not introduce too much computational effort into the algorithm.

Changing the height of the images also changes the height of the cells, which is going to be analyezed in the following section.

3.2.2 Number of Cells

Description of cells and analysis As mentioned before, each frame extracted is partitioned into a selected number of cells. These cells have the same width of the frame extracted and a variable height, which changes accordingly to the parameter *number of cells*. This parameter is directly related to the Normalization Height. Therefore, the height of the cells also determines the recognition rate. :

$$Number\ of\ cells = \frac{Height\ of\ the\ image}{Height\ of\ cells}$$

Changing the number of cells only affects two features: f2 and f15 (section 2.5), because these are the only features which take the cell-partitioning into account. Feature 2 gives the number of black/white transitions between two consecutive frame cells, while feature 15 provides the same information to the one that feature 2 does, with the only difference that feature 15 considers only the part above the lower baseline.

At first thought, we could arrive to the conclusion that the more cells we have, the more detail we will have about the frame extracted and that therefore, the result would be better. However, increasing the height of the image means increasing the number of cells, and as we saw in the previous section (3.2.1), high values for height normalization do not lead to good results.

Test Results Two different test sets were carried out to analyze the parameter *number of cells*:

- Tests with height normalized to 45 pixels.
- Tests with height normalized to 84 pixels.

Tests with height normalized to 45 pixels The best recognition rate was given with the parameter set to 21 cells, as it is shown in the Table 3.3 and Figure 3.1

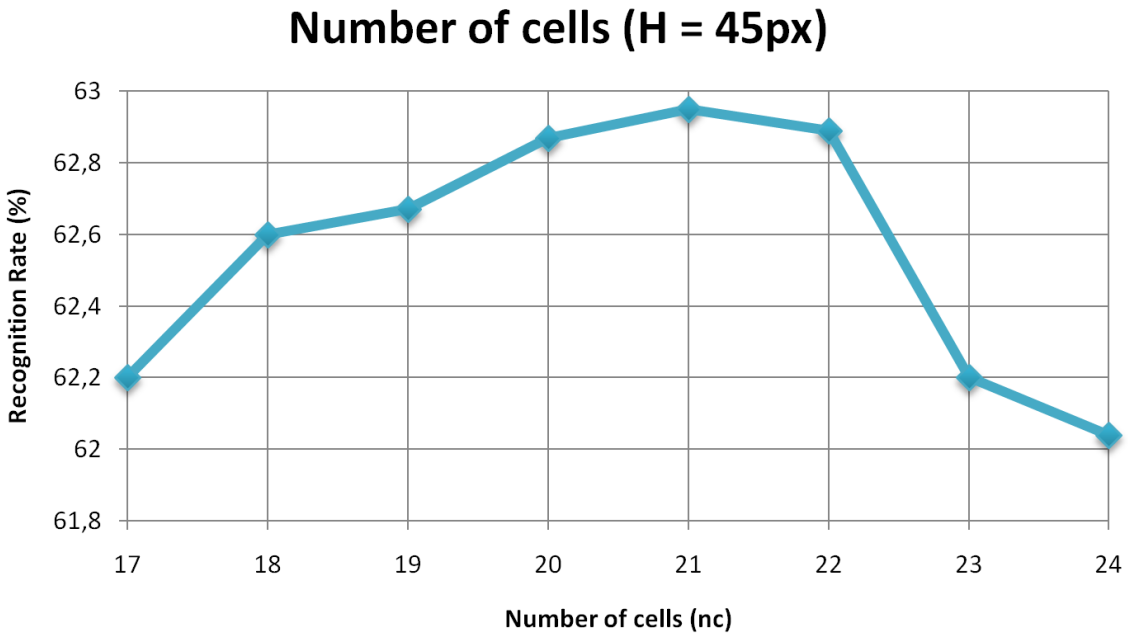


Figure 3.1: *Results with different number of cells. H=45 pixels.*

Number of cells	Height	Height/nc	Result
17	45 pixels	2,65	62,20%
18	45 pixels	2,50	62,60%
19	45 pixels	2,37	62,67%
20	45 pixels	2,25	62,87%
21	45 pixels	2,14	62,97%
22	45 pixels	2,05	62,89%
23	45 pixels	1,96	62,20%
24	45 pixels	1,88	62,04%

Table 3.3: *Results for different number of cells. H=45 pixels.*

Tests with height normalized to 84 pixels On this second test, the best recognition rate was achieved with the parameter *number of cells* set to 39. However, what it is striking about these test results is the ratio Height/nc. In both tests, the best results were achieved with the ratio Height/nc = 2,14 and 2,15. This demonstrates that the real parameter which is important to take into account is *Height/nc*, which corresponds to the height of the cells, and that the best results are given by values near to 2,15.

These same results also show that the recognition rate is never of a good value when the height of the cells is set to a value less than 2. If we make the height of the cells to be equal to 2 pixels, the previous equation looks like this:

$$Number\ of\ cells = \frac{Height\ of\ the\ image}{2}$$

Therefore, the number of cells must never be higher than the half of the height of the image.

3.2.3 Offset

As it was defined before, the *offset* is a parameter which determines the distance in pixels between two consecutive frames. The offset will vary the number of frames extracted and therefore, the amount of the information that will be introduced on the HMM system. The lower the offset, the higher the number of frames that will

Number of cells	Height	Height/nc	Result
37	84 pixels	2,27	59.87%
38	84 pixels	2,21	59.72%
39	84 pixels	2,15	60.12%
40	84 pixels	2,10	60.04%
41	84 pixels	2,05	59.54%
42	84 pixels	2,00	59.77%
43	84 pixels	1,95	57.69%
44	84 pixels	1,91	58.21%
45	84 pixels	1,87	58.51%
46	84 pixels	1,83	58.66%
47	84 pixels	1,79	58.95%
48	84 pixels	1,75	58.95%

Table 3.4: *Results for different number of cells. H=84 pixels.*

be extracted. For example, if an image is 400 pixels wide and the offset is set to 1 pixel, the frames extracted will exactly be 392. In contrary, if the offset is set to 3, the frames extracted will be exactly 132. Therefore, the difference between a good recognition rate and a bad recognition rate will be related to the amount of information that is introduced in the HMM system.

In all the tests performed, it has been proved that the best values for the overlap parameter are 2, 3 and 4. Especially, setting the value of the offset to 3 pixels has carried out the best results. Increasing the value of the offset means that fewer frames will be extracted from the image. Therefore, HMM systems will not have enough information for recognizing the objects. This effect can be clearly seen in the Figure 3.2 and Table 3.5, as the results for high offset values are very poor.

Number of cells	Offset	Height	Result
21	1 pixels	45 pixels	54,04%
21	2 pixels	45 pixels	64,00%
21	3 pixels	45 pixels	66,86%
21	4 pixels	45 pixels	62,95%
21	5 pixels	45 pixels	49,22%
21	6 pixels	45 pixels	32,97%
21	7 pixels	45 pixels	18,50%
21	8 pixels	45 pixels	9,67%

Table 3.5: *Results for different offset values.*

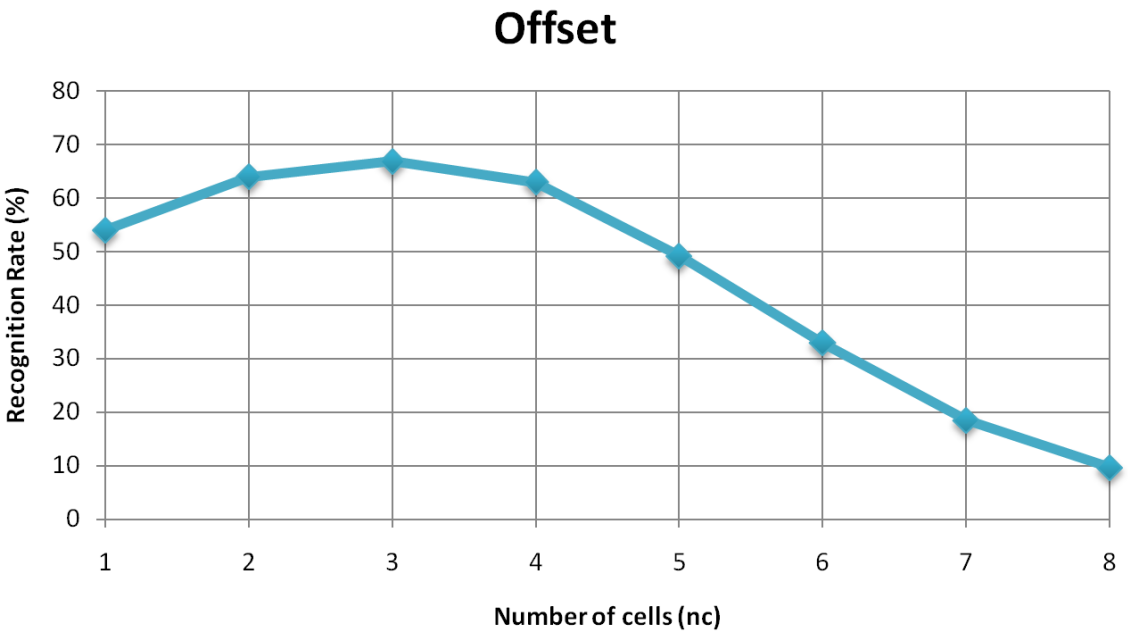


Figure 3.2: *Curve for the Overlapping Parameter.*

Clearly demonstrated by the results, the choice of the offset value has to be very carefully made, due to if a high offset value is selected, the recognition rate decreases considerably.

3.2.4 Alpha

Alpha determines the slant angle of the Sliding Window. If alpha = 0, the Sliding Window will be vertical, if alpha has a negative value, the Sliding Window will be inclined to the left and if alpha is positive, the Sliding Window will be leaned to the right. An example of this can be seen in Figure 2.6.

Tests In Figure 3.3 and Figure 3.4 the recognition rates for different angle values are shown. Each graph is accompanied by its corresponding data table.

In all the tests performed, it was demonstrated that the best recognition rates are obtained using vertical Sliding Windows ($\alpha = 0°$). The reasons for being the vertical window the best for feature extraction might be on one hand that most of the handwritten text images in the IFN/ENIT database are horizontal and therefore, the Sliding Window adapts better to the strokes if it is vertically oriented.

In conclusion, and especially for the IFN/ENIT database, vertical windows are proved to be the most effective. This does not mean that slanted windows are ineffective. In fact, it is going to be proved on section 4 that they are very useful when the combination scheme arrives, as they contribute additional information to the vertical window.

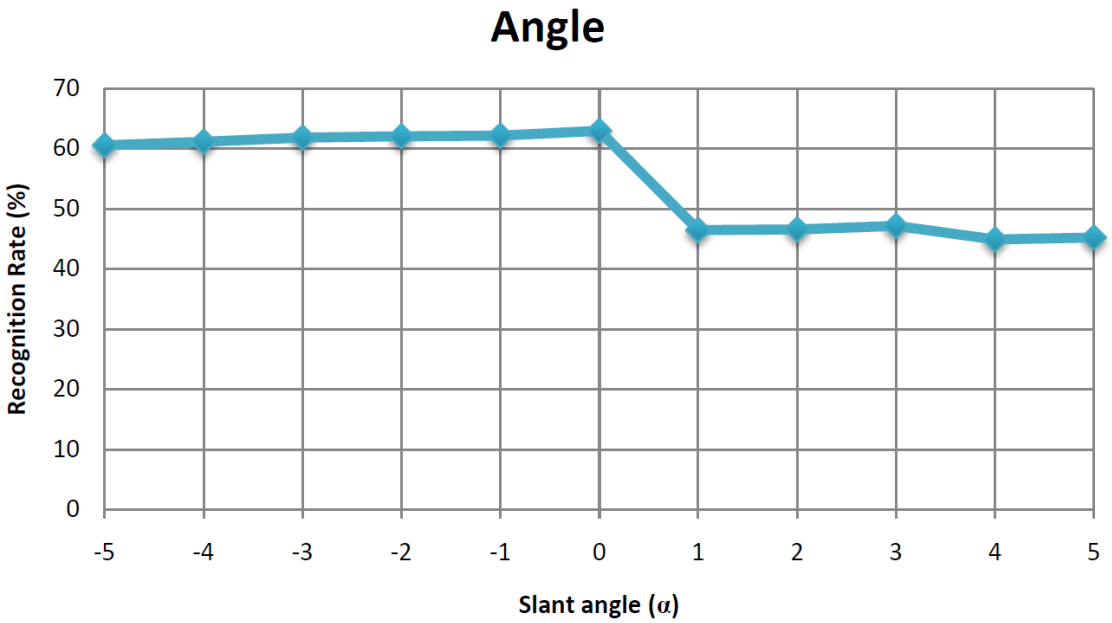


Figure 3.3: *Results for different slant angles for -5 < α < 5.*

Number of cells	Offset	Angle	Result
21	4	0	62,95%
21	4	-1	62,17%
21	4	-2	62,05%
21	4	-3	61,85%
21	4	-4	61,16%
21	4	-5	60,54%
21	4	+1	46,44%
21	4	+2	46,57%
21	4	+3	47,14%
21	4	+4	44,89%
21	4	+5	45,20%

Table 3.6: *Recognition rates for different angle values. Nc = 21, offset = 4.*

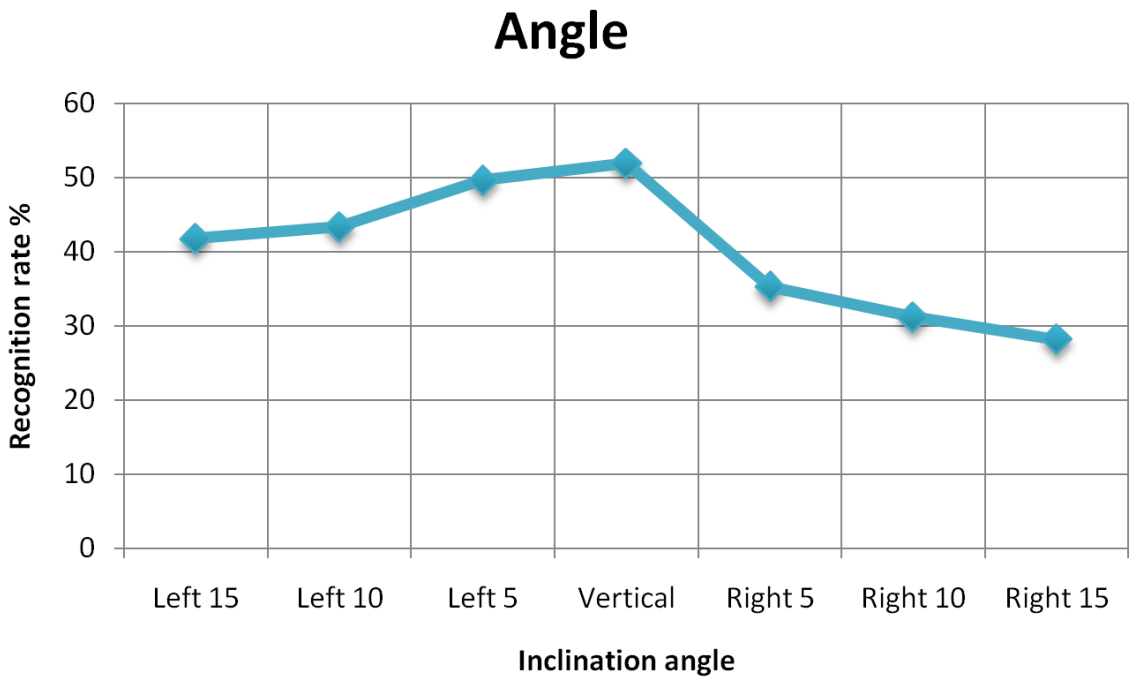


Figure 3.4: *Results for different slant angles for -15 < α < 15*

There are two main conclusions that can be stressed after having performed

Number of cells	Offset	Angle	Result
8	2	0	52,03%
8	2	-5	49,72%
8	2	-10	43,36%
8	2	-15	41,83%
8	2	+5	35,32%
8	2	+10	31,24%
8	2	+15	28,18%

Table 3.7: *Recognition rates for different angle values. Nc = 8, offset = 2.*

different tests changing the value of the frame slant angle:

Another striking conclusion is that negative values for *alpha* have been demonstrated to end up in better recongition rates than negative values. This simply occurs because there are more handwritten words inclined to the left side than the ones inclined to the right side.

3.3 Slant Correction

Considering that not all of the handwritten words are oriented 0° with respect the vertical, a slant correction method could be helpful for achieving a better recognition rate. The slant correction technique consists of calculating the stroke direction on each word image, and then using this particular angle for the Sliding Window. For example, if the stroke direction of an image is 4° to the right respect the vertical, then a sliding window slanted 4° would be used in this image. This way, a different value for the angle parameter would be used for each image.

The aim of this technique is to improve the results obtained by the vertical sliding window, making each image to be analyzed with a suitable Slanted Window.

The algorithm which recognizes the stroke direction in each image was designed using chain-codes. In this algorithm, the orientation of the borders of the handwritten text are analyzed. When the orientation of these borders are between the angles -45° and +45° or 225° and 315°, their directional values are saved in an array and finally, the final average angle value is calculated.

Tests Using the experimental setup described in section 3.1, two tests were performed to test the validity of the Slant Correction algorithm. In one of them the Slant Correction algorithm was used and in the other, the Sliding Window used was vertical. The results in Table 3.8 show that the slant correction algorithm designed does not provide good recognition rates.

Using Slant Correction	Using Vertical Window
42,49%	62,97%

Table 3.8: *Results with and without the Slant Correction algorithm.*

This results show that the Slant Correction step is not very recommendable to use. The main goal of this algorithm was to detect slanted handwriting. However, not all the inclined strokes in the handwritten words are caused by a slanted handwriting. What is more, the shapes which a word contain, may help the recognizer to detect a particular city name. Here, the Slant Correction may play a negative role cancelling these directional characteristics.

3.4 UOB System

The UOB system was presented at the ICDAR 2005 competition, where it carried out good recognition results. It basically consists of 24 of the 28 features defined on the original paper, leaving out the features f21, f22, f27 and f28 (section 2.5).

If compared to the original 28-feature set used in this work, its greatest advantage is that it does not require a big computational effort due to they are only 24 features.

Only a simple test was carried out in order to prove whether the recognition rates achieved by the UOB System differ significantly, and in fact, the results were not as good as the ones given by the original system. For these tests, the configuration defined in section 3.1 was used.

UOB System	Original System
62,90%	66,16%

Table 3.9: *Recognition Rate for the UOB System.*

For these tests, the following parameters were chosen:

4 Combinations

The main goal of this section is to show how the combination of different HMM recognizers can help in achieving a better recognition rate.

In section 2.3, the system used in this work was introduced and each one of its parts were explained. There, the Fusion Module was introduced as the part of the system which combines all the results given by different HMM classifiers, each one of them having a different result from a differently Slanted Window. These combinations can be performed in many different ways. In this section, the optimum values for the different parameters are chosen and introduced to new tests, where the Fusion Module is going to be used for the first time.

Two different algorithms were chosen to perform the Fusion Module. The first algorithm is called Simple Majority Vote, and the second one Feature Set Concatenation.

1. **Simple Majority Vote:** For each input image, each of the HMM classifiers give a Tunisian village name as a result. Therefore, we will have 3 city names per image. This algorithm consists on choosing the most repeated result between the 3 HMM classifiers. Sometimes, all of the HMM classifiers will give the same city name as the unique result and this is going to be the final answer. Other times, two of the results will be the same and the third will be different. In this case, the majority is going to be chosen as the final answer. In the third case, each answer is different from each other. In this latter case, the *dominant result* is going to be chosen as the correct answer. The *dominant result* will always be the result given by the vertical Sliding Window becasue, as demonstrated in subsection 3.2.4, the vertical Sliding Window is the one which carries out best results.

2. **Feature Set Concatenation:** When the 28-feature set is extracted from each frame, there are two possible ways for it. One possibility is to enter the HMM Classification Process, in order to get the result from the features. The other possibility is to concatenate the 28-feature sets given by different Slanted Windows with each other. This way, the combination process would be performed before the HMM Classification Process.

Tests Two definite tests were performed finally, using the optimum values for the parameters analyzed. These values are:

- Number of cells = 21.
- Offset = 3.
- Angles = 0 for the Vertical Window and -10° and +10° for the Slanted Windows.

Appart from that, the original shape of the handwritten text was kept to be the same, as it was discussed in section 2.8. Also, the algorithm for extracting the Upper Baseline was changed, and the method which carried out the best results in section 2.4 was chosen. This method increased in 5% the average density value for detecting the Upper Basline. To extract the Lower Baseline, the original method was used.

With all these considerations, these were the final recognition rates achieved:

- Using Simple Majority Vote: 71,27%.
- Using Feature Concatenation: 71,74%.

5 Conclusions

- The optimum parameter values given in this work should not be considered as definite values. A slight change in the feature extraction algorithm could change the recommended values and therefore, new tests should follow to any change in the algorithms.

- In order that the HMM classifiers can receive all the images under the same conditions, the standarization of the images is a very important process. This standarization helps the recognition process get better results. A clear example of this was the significant improvement introduced by the Height Normalization technique in subsection 3.2.1.

- The configuration of the HMM classifiers used in this bachelor's thesis was not the optimum, and could have been changed any time to obtain better recognition results. Therefore, the results given in this work should be observed as relative results and not as absolute results.

- Slanted Windows should never be performed individually in the Feature Extraction Process. As shown in subsection 3.2.4, low recognition rates are obtained when they are used individually. However, they are of a big use when they are combined with Vertical Windows, as they contribute with additional information that Vertical Windows cannot provide.

- Slanted Windows should never be performed individually in the Feature Extraction Process. As shown in subsection 3.2.4, they produce low recognition rates when they are used individually. However, they are of a big use when they are combined with Vertical Windows, as they contribute additional information.

- Slanted Windows with an angle between -10° and +10° provide good enough results individually. However, since Slanted Windows should provide additional information to Vertical Windows, the Slanted Windows with an angle lower than -10° or higher than +10°, provide a more useful information and make the Fusion Module to be more complete.

- In the original system explained in the paper [1], the Fusion Module was located and performed after the HMM Classifiers. However, the Feature Concatenation

method for combining the results was demonstrated to carry better results than the Simple Majority Vote (chapter 4). Therefore, the Fusion Module could be located before the HMM Classifiers.

Bibliography

[1] Ramy Al-Hajj Mohamad, Laurence Likforman-Sulem and Chafic Mokbel *Combining Slanted-Frame Classifiers for Improved HMM-Based Arabic Handwriting Recognition.* IEEE Transactions on pattern analysis and machine intelligence, vol.31, no.7 July 2009.

[2] Voker Märgner and Haikal El Abed *Pattern Recognition Systems. Conception, Evaluation and Improvement.* ICDAR 2009. July 2009.

[3] Hanan Aljuaid, Dzulkifli Mohamad and Muhammad Sarfaz *Arabic Handwritting Recognition Using Projectin Profile and Genetic Approach.* 2009 Fifth International Conference on Signal Image Technology and Internet Based Systems

[4] Nafiz Africa and Fatos T. Yarman-Vural *Optical Character Recognition for Cursive Handwritting* IEEE Transactions on pattern analysis and machine intelligence, vol. 24, no. 6. June 2002.

[5] Najoua Essoukri, Ben Amara and Faouzi Bouslama *Classification of Arabic Script Using Multiple Sources of Information: State of the Art and Perspectives* IJDAR 2003.

[6] M. Blumenstein, C.K. Cheng and X.Y. Liu *New Pre-Processing Techniques for Handwritten Word Recognition*

[7] Ramy El-Hajj, Laurence Likforman-Sulem and Chafic Mokbel *Arabic Handwriting Recognition Using Baseline Dependent Features and Hidden Markov Modeling*

[8] H.J. Kim, S.K. Kim, K.H. Kim, and J.K. Lee *An HMM-Based Character Recognition Network Using Level Building* Pattern Recognition, vol. 30, no. 3, pp. 491-502. 1997.

[9] Volker Märgner, M. Pechwitz and Haikal El Abed *Arabic Handwriting Word Recognition Competition* Proc. Eighth International Conf. Document Analysis and Recognition, pp. 70-74. 2005.

[10] M.A. Mohamed and P.D. Gader *Handwritten Word Recognition Using Segmentation-Free Hidden Markov Modeling and Segmentation-Based Dynamic Programming Techniques* IEEE Trans. Pattern Analysis and Machine Intelligence, vol. 18, no.5, pp. 548-554. May 1996.

[11] M. Pechwitz and V. Märgner *HMM-Based Approach for Handwritten Arabic Word Recognition Using the IFN/ENIT-Database* Proc. Seventh International Conf. Document Analysis and Recognition, pp. 890-894. 2003

[12] A. Vinciarelli and J. Luettin *Off-Line Cursive Script Recognition Based on Continuous Density HMM* Proc. International Workshop Frontiers in Handwriting Recognition, pp. 493-498. 2000.

[13] Pattern Recognition
`http://en.wikipedia.org/wiki/Pattern_recognition`

[14] MATLAB
`http://en.wikipedia.org/wiki/MATLAB`

[15] Image Processing Toolbox
`http://www.mathworks.de/products/datasheets/pdf/image-processing-toolbox.pdf`