Journal of Computer Science & Technology, vol. 6, 2006, pp. 92-98.

Manuscript Character Recognition. Overview of features for the Feature Vector.

De Giusti, Marisa, Vila, María Marta y Villarreal, Gonzalo Luján.

Manuscript Character Recognition
Overview of features for the Feature Vector

Marisa R. De Giusti, María Marta Vila, Gonzalo Luján Villarreal

Comisión de Investigaciones Científicas de la Provincia de Buenos Aires (CIC)
Proyecto de Enlace de Bibliotecas (PrEBi)
Servicio de Difusión de la Creación Intelectual (SeDiCI)
Universidad Nacional de La Plata
La Plata, Argentina

marisa.degiusti@ing.unlp.edu.ar, {vilamm, gonetil}@sedici.unlp.edu.ar

ABSTRACT

The image recognizing process requires the identification of every logical object that compose every image, which first implies to recognize it as an object (segmentation) and then identify which object is, or at least which is the most likely one from the universe of objects that can be recognized (recognition). During the segmentation process, the aim is to identify as many objects that compose the images as possible. This process must be adapted to the universe of all objects that are looked for, which can vary from printed or manuscript characters to fruits or animals, or even fingerprints. Once all objects have been obtained, the system must carry on to the next step, which is the identification of the objects based on the called universe. In other words, if the system is looking for fruits, it must identify univocally fruits from apples and oranges; if they are characters, it must identify the character "a" from the rest of the alphabet, and so on.
In this document, the character recognition step has been studied. More specifically, which methods to obtain characteristics exist (advantages and disadvantages, implementations, costs). There is also an overview about the feature vector, in which all features are stored and analyzed in order to perform the character recognition itself.

1. PRE PROCESSED IMAGES

Images that we are dealing with generally contain different levels of noise that can affect the information obtained from them. Moreover, this images contain unnecessary information or information that makes harder to make an accurate feature extraction. This information is not always noise but flaws that belong to the original image and which must be eliminated or minimized to permit the processing of the image itself. The pre processing stage consists in minimization or elimination of these variables, which can be different line width, slant correction, base line detection and correction, size character differences, gray levels, and so on.

Once the first stage has finished, all images have been normalized, resulting in clearer stood-out relevant information new images [2], [3], [4].

2. OBJECTS INSIDE THE IMAGE

A segmentation process must be performed over normalized images in order to obtain all logical objects that compose every image.

2.1 Images composed by objects.

Once images have been normalized, a segmentation process must be performed in order to obtain the objects that compose the target image (or at least, as many as possible). Segmentation methods do not always obtain the right objects, or maybe two or more objects are interpreted as a single one. Although preprocessing stage tends to minimize this kind of mistakes, many times some objects are still too hard to identify.
This situation is very common in handwritten text recognition, where characters are either too close or bad formed, generating objects that seem like other characters. These wrong characters, once in recognizing stage, might be recognized as a different character or not recognized at all, since the information obtained will not match with any pre established pattern.
The whole situation would be more confusing if images are mixed with text, such as inside maps or tables, since this implies first to separate text from images, and then the segmentation of the text itself.
There exist many segmentation methods each one adapted to the problem the application is trying to work out. Some methods highlight borders in the images, others try to find some pattern on every object, others operate over histograms, and so on. Nowadays it is very common to use a dictionary together with the recognizing process, which allows the search of similar words and error correction during recognizing process. Interested readers may get more information in [1], [5], [6], [7], [8].

2.2 Objects Recognition. Feature Extraction

Once objects have been obtained, it is time to recognize them. So the software will try to obtain as many information as possible from the objects, and then to compare with the models stored until the one that maximize the coincidence is found. This comparison can be made using neural networks, probabilistic process, Hidden Markov Models, and many others. Although some methods are more complex than others, they often present many advantages. For instance, with neural networks the program can train itself and learn, which will eventually make recognition more accurately adapting itself to the objects in order to increase the probability of finding an object.

Those methods are based in a set of values of certain features obtained from the objects. This process is known as **feature extraction**, and consists in obtaining of as much information as possible from the object, whether from the object itself, related to other object or even the whole image, in order to make an univocal characterization of the object.

During the feature extraction process, every object is analyzed and all values from the predetermined set of characteristics are obtained. This set is different for each kind of problem that any recognition software is made for. In the case of manuscript character recognition, the size of this set might be very large, but this does not necessarily improve the whole system. There are two factors that determine the right size for this features set:

1. Quantity - execution time relationship
2. How relevant and how much every feature contributes

The first point is a determinant in this process, although logic would indicate that the more features are extracted the more exact would be the recognition, this is not always like that. The problem arises because if more and more features are added to the set, the execution time increases considerably.

The second point is very linked to the first one, and it indicates that not all features contribute with relevant information, or at least they do not increase the amount of matches.

Sometimes it is convenient to do without some features that do not contribute considerably, and to keep those that are really worthwhile to be kept, in order to increase the recognition process speed eliminating unnecessary process.

A third point comes out here, partly related with the relevance of the feature, that is the importance of every feature. Although much information can be obtained from the images, not every data has the same importance. The recognition system associates all features with a value that detonates its importance, which is named weight. All weights are stored in a vector, named weights vector (W); every position i in the vector W indicates the weight of the feature W_i. In the same way, after analyzing and obtaining the values of every feature for each object in the image, these values are stored in a vector of values V, which will have the same size that vector W. Thus, the result to consider for the system from a particular feature C_i of a particular object will be determined by its weight and the value obtained. This is written as:

$$Oc_i = W_i . V_i$$

which is equivalent to

$$Oc_i = W[i] \times V[i]$$

Where:

O is the object being analyzed
C is the features vector
W is the weights vector
V is the vector of values obtained from the image
i is the position in the feature vector

Some features are based in the objects, while others are based in the objects and their relationship with their environment (for instance, comparing height and width between objects).

Up to here, it has been explained what the features vector and the weights consist of. Coming next, some methods will be analyzed and as well as some useful features related with the problem.

2.3 Base Line, Descendents and Ascendants

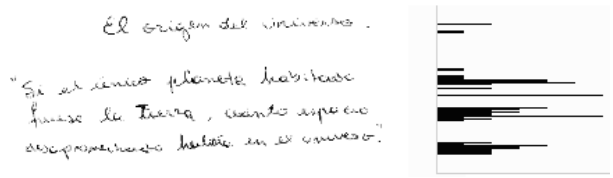
The base line can determine where the line of writting is. In other words, it allows to find a straight line which may determined the central position of the text. This feature is very useful to find ascendents and descendents of every word, which means characters mostly *height* (or higher than most characters) and characters mostly *shorts*. Typical cases in manuscript handwritten are:

Ascendents: letters “l”, “h”, “k”, “d”
Descendents: letters “j”, “g”, “p”, “q”
Both: “f”
None of them: “a”, “c”, “m”

This technique is very useful to make a classification of characters in 4 different groups; it is important to make clear that finding out that a character accomplishes to some groups characteristics does not mean that the character belongs to the group, but it only increases the probabilities of finding a determined character.

There exist many ways to find the base line; a very

useful technique is to use horizontal histograms, calculating the amount of black pixels in each horizontal line. Then, an analysis is made over that histogram, looking for maximums and minimums both locals and absolute. Most dense zones indicate the center of the line, while less dense zones indícate ascendents and descendents in the line.
This technique has many other uses in line-segmentation steps; by performing an analysis over the horizontal histogram from the whole image (Image 1.a), and considering local minimums and maximums as line break.



(a) Histogram from the whole image

(b) Histogram of one word

Image 1 – Histograms

Once the histogram has been obtained, it is relatively easy to know if a character exceeds the medium of all characters, or if its position is over or under de line (compared with other characters). If a technique such as Word Spotting is being used, where the word is treated as a whole (Image 1.b), it is possible to know the amount of ascendents and descendents of a single word, and their position in the word itself (in the middle, front, back, first quarter, …). These methods have proved to be very useful and effective, resulting in a viable alternative. [10].

2.4 Holes and Classification

To know the amount of holes that an image has could be very useful, delimiting the characters to recognize. We count again with several subsets of characters, this time clasified by the amount of holes:
0 hole: 1,2,3,..,c,f,h,...
1 hole: a,b,d,o
2 holes: 8,B

With some letters, for instance g letter, it depends the way they are written the amount of holes they will have (it could be both 1 or 2 holes). This kind of issues is very common when dealing with handwritten text, and they must be considered.
In order to find holes inside the images, a very spread and useful method is based con Connected Components. As it has been explained in [2], connected components can be calculated for white (background) or for black (foreground, the ink itself). When calculating connected components for the background of the image, what is actually being done is identifying zones entirely divided by a character. One zone belongs to the background, and the rest to the image generated from the character. Thus, the amount of holes in an image CA will be:

$$CA = CC - 1$$

where CC is the amount of Connected Components.

These holes can be classified according to their height and width. This classification assumes that information about each connected component has been stored in some kind of data structure, which would be later processed.
This may be achieved by using a grid with the same size of the target image, in which connected components will be labeled while they are being calculated (some methods can do it automatically). Then, these labels can be analyzed in order to find the information required.
The knowledge of the holes in the image contributes with very important information, but to process and classify these holes do not add relevant information; in addition they are very time consuming.

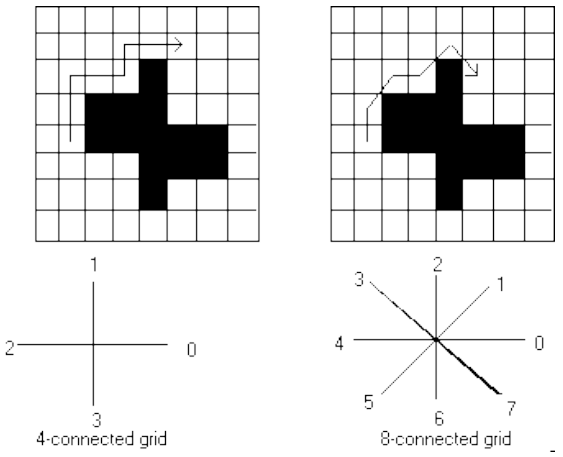


Image 2 – Connected components for the same image

3. CHAIN CODES

The chain code is a technique widely used in many fields of image processing (geography, archeology,

and many more); with this technique we can obtain a path over the image skeleton or structure, and then know a lot of information from this path, such as changes in the direction, cycles, length of the stroke, and so on. A great advantage that this technique offers is that it can be used in image segmentation as well as feature extraction and classification.

3.1 Chain Codes Calculation

To calculate the chain code, it is often used a numbered scheme, where each numeric value represents one direction; it can be used both 4-connectivity and 8-connectivity. Which type of connectivity we shall choose will be determined by the method used to extract the skeleton of the image (since we are already dealing with the image skeleton; see [2]) as well as how exactly we want the chain code to be.
First, we must go along the image skeleton, starting by the left-lowest pixel, registering the direction of the path, and setting every pixel as *visited* (in order to avoid infinite cycles).

Among many applications of this technique, one very useful is to find maximum curvature points inside the chain code. Two different ways to do this is:

> Based on each pixel coordinate in the chain code, calculate the tangent vector. [11] [12] [13] [14]
> Estimate the tangent in a point in the chain code by using partial derivates in the image. [15]

This technique is also very useful to find valleys and hills, which can be done by detecting abrupt changes in the path. For instance, if the chain code is 77777771111111 then we can say that we are seeing a very deep valley (such as a V letter). It is convenient to keep in consideration the length of the lines, as well as the amount of valleys and hills inside the image.

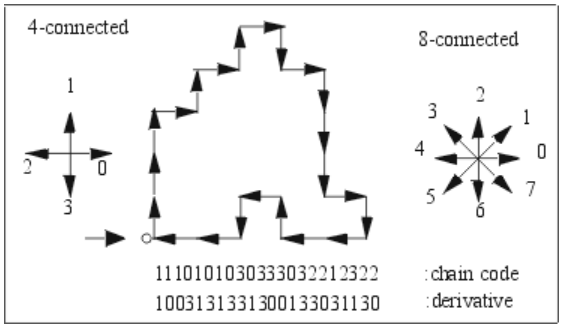


Image 3 – Chain code calculation

The Chain Code contributes with a lot of information, and has an acceptable cost in execution time. All the information taken with this technique is very important and considered very relevant. It is important here to make clear that the chain code is just a method, and not a characteristic to be stored in the feature vector; it is useful to take features from the image in order to store them in the feature vector.

As before mentioned, the chain code is based on the skeleton of the image. We have observed that depending the method used to get the skeleton, results might be more or less accurate. In particular, based on some tests we have done (see [2]), it has been observed that the use of connected components could be better to find the chain code. Although the cost to obtain connected components might be considerably high, results may be reused to classify holes, which combined with chain codes may result in a hybrid method very powerful, efficient and robust.

4. FEATURES BASED ON SAMPLE WINDOWS

The image to process is divided using an AxB matrix, with squared cells (same width and height).

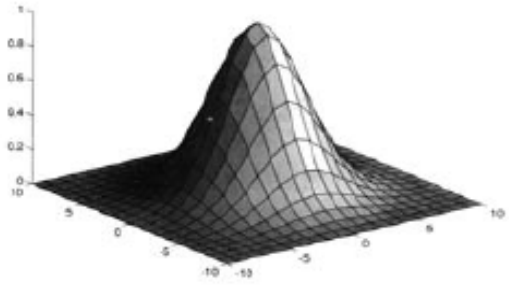


Image 4 – Gaussian filter

Then, over each cell, a squared zone is selected. This zone is known as the sample window, and may cover totally or part of the neighbor cells where the window is centered. A process is performed over each window, obtaining many useful and interesting features, such as gray level, horizontal and vertical derivates, local slope and correlation coefficient.

4.1 Grey Level

To calculate the grey level on each sample window, it must be analyzed how much grey adds every sample point of the window. But not every point contributes in the same way, since points near de

center of the window are more important than the ones near the edges. In order to represent this, a Gaussian filter is applied centered on the sample window, and then the grey intensity $I_{x,y}$ is calculated over each point in the window (x,y).
The Gaussian filter is a linear filter with the following function:

$$g(x) = e^{-\frac{x^2}{2\sigma^2}}$$

with zero mean and standard deviation σ. Variable x represents the simple point. This implies that, when dealing with 2D images in a plane, x = (i,j), and thus

$$g[i,j] = e^{-\frac{(i^2+j^2)}{2\sigma^2}}$$

The Gaussian filter is applied using a squared mask of NxN (we chose a 5x5 mask) with values that tends to Gauss bell. Clearly, when N is big, more exact would be the approximation, but again the time needed to process the mask will be considerably increased.

0.00	0.01	0.02	0.01	0.00
0.01	0.06	0.10	0.06	0.01
0.02	0.10	0.16	0.10	0.02
0.01	0.06	0.10	0.06	0.01
0.00	0.01	0.02	0.01	0.00

Image 5 – Gaussian filter grid

To find $I_{x,y}$ the next formula is used:

$$\hat{I}(x,y) = I(x,y)\exp\left[-\frac{1}{2}\left(\frac{x-(n/2)^2}{(n/4)^2}+\frac{y-(m/2)^2}{(m/4)^2}\right)\right]$$

Where:

n is the width of the sample window
m is the height of the sample window

After calculate all grey intensities for each point in the sample window and add them, a value for the grey intensity of the whole window is obtained, which is later normalized between 0 and 1. This result is registered and the process carries on with the next sample window.

4.2 Horizontal and vertical derivatives

If lines are treated as mathematical functions, they can be analyzed looking for slopes, and grow and decreasing zones.
For each sample window, horizontal and vertical derivatives are calculated, both based on points in the window itself. This process is alike in both cases. In case of horizontal derivative, the process is as follows:

$$g_i = \frac{\sum_{j=1}^{m} I(x_i, y_j)}{m}$$

- the average grey level is calculated for each column in the window.
- the straight line $\mathbf{ax_i + b}$, which distance to the points (x_i,g_i) minimizes the function J, is searched:

$$J = \sum_{i=1}^{n} w_i (g_i - (ax_i + b))^2$$

Where:

a is the slope of the line
b is the y-intercept
x_i column of the sample window
n amount of columns
g_i gray level previously calculted
wi Gaussian filter function for x_i:

$$w_i = \exp\left(-\frac{1}{2}\frac{(x_i - n/2)^2}{(n/4)^2}\right)$$

To find this straight line ($ax_i + b$) requires linear regression. The value "a" is taken as the value of the horizontal derivative of the sample window. In order to calculate its value, together with b, of the line which minimizes J, partial derivatives must be annulated, for a and for b:

$$a = \frac{\left(\sum_{i=1}^{n} w_i g_i\right)\left(\sum_{i=1}^{n} w_i x_i\right) - \left(\sum_{i=1}^{n} w_i\right)\left(\sum_{i=1}^{n} w_i g_i x_i\right)}{\left(\sum_{i=1}^{n} w_i x_i\right)^2 - \left(\sum_{i=1}^{n} w_i\right)\left(\sum_{i=1}^{n} w_i x_i^2\right)}$$

The same procedure must be followed to calculate the vertical derivative, switching rows by columns, x_i by y_i and x by y. It is convenient to convert results in radians when dealing with vertical slopes (between [-л/2, л /2]), since it will ensure that we are in a linear and delimited range. This is specially useful in cases of slopes close to the vertical.
Techniques based on sample window permit to obtain better results if during the segmentation stage all logical objects (or at least as many as possible) have been extracted. Anyway, they are by themselves a very robust method. Clearly, its robustness required a high processing cost that must be paid, and specially in large images.

4.3 Invariant Features

Geometric distortions of letters and words (rotation, scale, translation) are a problem that must be faced no matter which feature we are working with. In some cases, it may be minimized if a normalization process is applied before, as it has already been mentioned, but this would increase the processing time for the algorithms; besides it could provoke a lost of information and the risk of taking premature decision based on inclomplete information. To avoid this kind of problems, there exist some methods which allow to obtain features but that do not depend on the variability to make the pattern recognition. Those methods go often together with other methods (as the ones already mentioned), in order to obtain raw features first, and features based on normalized information.
In [17] two features are introduced, which are invariant to geometric distortions: *Normalized curvature* and *Signed Tangent Coefficient*.

Normalized curvature.

It is known as affine transformation as the transformation that results of the combination of **translation, rotation and scale,** defined by :

$$w = c\,\mathbf{U}\,r + v = c \begin{matrix} \cos w & \sin w \\ \cos w & \sin w \end{matrix} r \quad V_x \quad V_t^{\ T}$$

where c is a scalar positive.

Two curve segments are equivalents if they can be obtained one from the other using an affine transformation.
Let P(t) = (x(t),y(t)) be a planar smooth curve, which is mapped to $\widetilde{P}(\tilde{t}) \quad (\tilde{x}(\tilde{t}), \tilde{y}(\tilde{x}))$ by a reparametrization $t(\tilde{t})$ and an affine transformation, for instance P(t) = c **U** P($t(\tilde{t})$) + v . It is taken a reparametrization of both curves by the length of the arc, such as t = s y $\tilde{t} \quad \tilde{s}$. From here, it is inferred that $\widetilde{ds} \quad c.ds$, so points corresponding to both curves are related by $\widetilde{P}(\tilde{s}) \quad cUP((\tilde{s} \quad \tilde{s}_0)/c) \quad v$.It can be proved that the curvature in the corresponding points of two curves is scaled by 1/c (for instance, $\widetilde{K}(\tilde{s}) \quad \frac{1}{c}\widetilde{K}((\tilde{s} \quad \tilde{s}_0)/c)$). From here, we can now deduce the following equality: $\widetilde{K}(\tilde{s})\widetilde{K}^{\,2}(\tilde{s}) \quad K'((\tilde{s} \quad \tilde{s}_0)/c)K^{\,2}((\tilde{s} \quad \tilde{s}_0)/c)$, where $K' \quad dK/ds \; \widetilde{K} \quad d\widetilde{K}/d\tilde{s}$ and $K' \quad dK/ds$. This equation defines the **Normalized Curvature**.

Signed Tangents Coefficients.

This feature is based on the invariability of distance coefficients between two corresponding points. Let's again consider the two equivalents curves P(t) and $\widetilde{P}(\tilde{t})$ that have been defined before; let P_1 and P_2 be two points of the curve $P(t)$ and the slope of its tangent lines (T_1 and T_2 respectively) differs in an angle ; the point P is defined as the intersection between straight lines T_1 and T_2. It is define alike $\widetilde{P}_1$ and $\widetilde{P}_2$ for curve $\widetilde{P}(\tilde{t})$ and the slop of its tangents ($\widetilde{T}_1$ and $\widetilde{T}_2$ respectively) also differs in ; let's now define $\widetilde{P}$ as the intersection point between tangents $\widetilde{T}_1$ and $_0$ on $\widetilde{P}(\tilde{t})$. Since angles, and then changes in the curvature, are invariant to affine transformations, it can be demonstrated that point $\widetilde{P}_1$ corresponds to point P_1, and therefore points $\widetilde{P}_2$ and $\widetilde{P}$ correspond to points P_2 and P respectively. This means that $\left|\widetilde{P}_1\widetilde{P}\right| \quad c\left|P_1P\right|$ and that $\left|\widetilde{P}\widetilde{P}_2\right| \quad c\left|PP_2\right|$. From here we can say that:

$$\frac{\left|\widetilde{P}\widetilde{P}_2\right|}{\left|\widetilde{P}_1\widetilde{P}\right|} \quad \frac{\left|PP_2\right|}{\left|P_1P\right|}$$

The value of the angle must be fixed in $_0$. Thus, this equation defines a new invariant feature that can be computed for each point of the sample, and is known as *tangent coefficients.* The distinctive power of this feature can be yet improved if this coefficient is increased by the sign of the local curvature. This finally results in what is called **Signed Tangent Coefficients.**
Due to the methodology to process all objects used by this techniques, a previous thinning stage will make all the information obtained more useful (which again will increase the total time of the whole system)
On the other hand, one of the main advantages is that the segmentation stage requires only a segmentation by line and by word, since this technique permit to process words as a whole, studying curves that form a complete line. All this result in compensation in processing times, taking advantage of the goods that these techniques have to offer.

5. REFERENCES

[1] F. Wahl, K. Wong and R. Casey. "Block segmentation and text extraction in mixed text/image documents". *Computer Vision Graphics and Image Processing*, 20:375-390,1982

[2] Marisa R. De Giusti, Maria Marta Vila, Gonzalo Luján Villarreal. "Manuscript Document Digitalization and recognition: a first approach". *Journal of Computer Science and Technology*. Vol 5. No. 3. 158-163,2005.

[3] William Prat, John Wiley & Sons, Digital Image Processing, 1991, Second Edition

[4] Gonzalez Rafael, Woods, Addison-Wesley Digital Image Processing, 1992, Second Edition

[5] Michel Weinfeld, "Reconnaissance d'ecriture manuscrite: segmentation de mots". Département d'Enseignement et de Recherche en Informatique. École Polytechnique, Paris, France

[6] L. Fletcher and R. Kasturi. "A robust algorithm for text string separation from mixed text/graphics images". *IEEE Transactions on Pattern Analysis and Machina Intelligence*, 10:910-918. 1988

[7] R. Manmatha and Nitin Srimal. "Scale Space Technique for Word Segmentation in Handwritten Documents". *Computer Science Departament, University of Massachusetts*

[8] R. Manmatha, Chengfeng Han, E. M. Riseman and W. B. Croft. "Indexing Handwritting using Word Matching". Center for Information Retrieval, Computer Science Departament, University of Massachusetts

[9] Nicolas Nonmarché. "Algorithmes de fourmis artificielles: applications à la classification et à l'optimisation", 2000. Université Francois Rabelais Tours.

[10] R. Manmatha, Chengfeng Han and E. M. Riseman. "Word Spotting: A New Aproach to Indexing Handwriting". Center for Intelligent Information Retrieval, Computer Science Departament, University of Massachusetts.

[11] H. Asada y M. Brady. The curvature primal sketch. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 8(1):2-14, 1986

[12] M.H. Han, D. Jang, y J. Foster. Identification of corners points of two-dimensional images using a line search method. *Pattern Recognition*, 22(1):13--20, 1989.

[13] G. Medioni y Y. Yasumoto. Corner detection and curve representation using cubic B-splines. *Computer Vision, Graphics, and Image Processing*, 39:267--278, 1987.

[14] B. Bell y L.F. Pau. Contour tracking and corner detection in a logic programming environment. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 12(9):913--917, September 1990.

[15] R. Deriche y O.D. Faugeras. 2d curve matching using high curvature points: Application to stereo vision. In *Proc. of the 10th. International Conference of Pattern Recognition*, pages 240--242, 1990.

[16] Alejandro H. Toselli. Tesis doctoral: Reconocimiento de texto manuscrito continuo.. Universidad de Valencia. 2004

[17] Jianying Hu, Michael K. Brown, William Turin. HMM Based On-Line Handwriting Recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*. 10 (18):1039-1045. Oct. 1996