MANUSCRIPT DOCUMENT DIGITALIZATION AND RECOGNITION: A FIRST APPROACH

Marisa R. De Giusti[1], Maria Marta Vila, Gonzalo Luján Villarreal

Proyecto de Enlace de Bibliotecas (PrEBi)
Servicio de Difusión de la Creación Intelectual (SeDiCI)
Universidad Nacional de La Plata
La Plata, Argentina.
marisadg@ing.unlp.edu.ar, {vilamm, gonetil}@sedici.unlp.edu.ar

Abstract.
The handwritten manuscript recognizing process belongs to a set of initiatives which lean to the preservation of cultural patrimony gathered in libraries and archives, where there exist a great wealth in documents and even handwritten cards that accompany incunabula books. This work is the starting point of a research and development project oriented to digitalization and recognition of manuscript materials. The paper presented here discuss different algorithms used in the first stage dedicated to "image noise-cleaning" in order to improve it before the character recognition process begins.
In order to make the handwritten-text recognition and image digitalization process efficient, it must be preceded by a preprocessing stage of the image to be treated, which includes thresholding, noise cleaning, thinning, base-line alignment and image segmentation, among others. Each of these steps will allow us to reduce the injurious variability when recognizing manuscripts (noise, random gray levels, slanted characters, ink level in different zones), and so increasing the probability of obtaining a suitable text recognition. In this paper, two image thinning methods are considered, and implemented. Finally, an evaluation is carried out obtaining many conclusions related to efficiency, speed and requirements, as well as ideas for future implementations.
Keywords: patrimonial conservation, digitalization, thinnig, connected components

1. INTRODUCTION.

During the last years, the digitalization of document pieces watched over in libraries and archives of all over the world has attained a great significance for its infinite possibilities, both for specialists and a broader public. This new social space, enabled by digital technologies, allows us to discover a set of objects which constituted a highly unknown cultural richness. However, the current advantages of this technology have also represented a hard learning, which has encompassed dramatic mistakes paid with the destruction or considerable damage of original documents. For this practical reason, the problem of digitalization, and its own technological nature, has opened an international reflection space which is mainly characterized by the constant flow of specialized information and by the introduction of a constant social benefit discourse.
When digitized, documents are better preserved, though the search and access to such information is rather slow, sequential, and thus highly inefficient. Cataloguing such documents may solve the problem, but requires a large quantity of human and temporal resources to carry out this process; the automatic indexation of digitized documents - using both pattern recognition systems and database management systems - allows accelerating the cataloguing process as well as the searching and access processes, turning it into a viable solution in terms of complexity, times and costs.
In order to recognize manuscript handwriting in a digitized document, a set of operations allowing the individual characterization of each object, extracting the biggest possible quantity of characteristics, should be carried out. In order to make this process of feature extraction as efficient as possible, a previous processing of the image must be carried out, removing any ambiguous or confusing component among similar objects. One of the most important tasks is obtaining the skeleton of the objects, which is achieved by thinning such objects, keeping its size and shape.

2. PREVIOUS DEFINITIONS.

An image can be viewed as a set of points which have a certain value indicating its color[2]. In grayscale images, those values range from 0 (black point – without brightness) to 255 (white point).
The images we will work with have already been binarized (thresholded), applying to them a previous treatment in which, based on a threshold value, all the pixels surpassing such value are assigned 0, and 1 is assigned to those under such threshold. The new image constituted by 1s and 0s allows us to work in handwriting recognition eliminating the problem of colors or grayscale. The selection of the threshold can be manual or automatic, existing several methods for choosing the best threshold. Even though a detailed description exceeds the purposes of this paper, interested readers may consult [11].
The set of pixels (also called point $p = (x,y)$ with $x , y \in N$) making up an image can be considered in two ways: [3]
a) As a matrix, where $I [p] = I [(x,y)] = k$; with k pixel value x,y ($k \in [0,255]$ in gray images; $k \in [0,1]$ in binarized images)
b) As a function Pixel: (Image, point) -> value where Pixel(I , p) = Pixel(I , (x,y)) = k.

In order to **thin an image keeping the topology** (so as to not alter the shapes), the points belonging to the border should be removed, i.e. those points with the following properties:

[1] Investigador Comisión de Investigaciones Científicas de la Provincia de Buenos Aires – CIC.

[2]For colored images, each point has 3 values: red, green, and blue value (RGB). Interested readers may consult [10][11].
[3]The present work opts for the functional notation, using mathematical expressions and, in some instances, including matrices in the cases of concepts such as neighborhood and adjacency.

1) Black point (it is useless to remove white points)
2) Simple point
3) Final point exclusion
Where:

- A pixel P is considered as *simple point* if the set of P's black neighbors has **a single connected component** adjacent to P.
- A pixel P is considered as *final point* if it has a single black neighbor (end point of the image).

In order to eliminate the border points, several runs shall be carried out (thus keeping the topology). In each run, all the "removable" points should be marked, and then the removal process should be fulfilled. Runs will be repeated until the removal of at least one pixel is achieved.

Neighborhood Concept. Pixel's neighbors are conditioned to the type of meshing used for representing the image. q-neighborhood is defined (also q-adjacency) as a set of pixels neighbors of p. The value of q will depend on the type of meshing. If a hexagonal mesh is used, each image pixel (excluding the borders) has 6 neighbors, in which case we are dealing with a 6-neighborhood. A quadrangular mesh presents two possible options: a 4-neighbor (considering the pixels at the sides, up and down of the pixel, as neighbors) or a 8-neighborhood (idem 4-neighborhood plus 4 pixels of the diagonals).
(p,q) adjacency is defined as:
p-adjacency for black pixels
q-adjacency for white pixels[4]

Computation of connected components in an image. A connected component (CC) of an image is a set of pixels which fulfils the norm that, for each pair, there exist a digital path linking them. A **digital path** of a pixel p to another pixel p is a set of pixels $P_{pq} = \{p_i \; i=0..n\}$ so that:
1) $\forall p_i \in Ppq$, $color(p_i) = color(p) = color(q)$
2) p0 = p; pn = q
3) For all i=1,...,n-1, pi has exactly two neighbors in P_{pq} which are pi-1 , pi+1
4) p_0, p_n have exactly one neighbor which are p1 and pn-1, respectively.

Finally, a connected component is **bounded** if it does not have any border pixel.

3. AN ALGORITHM FOR COMPUTING CONNECTED COMPONENTS (4-ADJACENCY IS USED).

Considerations: the image used has been binarized. Pixels value 1 (black) or 0 (white). A matrix is used, of the same size of the image which will store labels to characterize the original image pixels according to its type of equivalence.
An image is traced from left to right and upside down. For each pixel p=(x,y), the neighbors p1=(x-1,y) and p2=(x,y-1) are examined.

- If they both value 0 (are white), a new label is created and is assigned to P in the label matrix.
- If only one is 0, the same label of the other (which is not 0) is assigned to P.
- If none is 0, the label of any of them is assigned to P. Two scenarios may be presented:
    - o The label s of p1 and p2 are equal, with which the label of p will be the same in both.
    - o The labels of p1 and p2 are different, in this case p receives the label of any of the two. Let's suppose it is assigned the label of p1. Then, it shall be recalled that, even though p2 and p have different labels, they belong to the same component.

If the labels are considered as types of equivalence to which pixels belong, if p1 belongs to type 1 and p2 to type 2, and during the image processing p1 and p2 belong to the same component, the type of equivalence of p1 is thus the same as that of p2. In consequence, Type 1=Type 2. In order to keep record of this, a vector in memory is used, which has as many places as labels used. Labels are integers, all of which allows us to use them to access directly the vector as indexes.
Each position of the vector will represent the label, and its content will represent the type of equivalence to which such label belongs. For example, if we have the vector:

V = {1,2,3,2,3,1,4}

the first position of the vector is position 0, the last is the n-1 (in our case, 6), and the semantics of vector V is: "*label 'x' belongs to the type of equivalence V[x]*", then:

- Labels 0 and 5 belong to the type of equivalence 1.
- Labels 1 and 3 belong to the type of equivalence 2.
- Labels 2 and 4 belong to the type of equivalence 3
- Label 6 belongs to the type of equivalence 4.

All of which means: let Eq :: Label -> Integer be a function taken by a label and returns the type of equivalence to which such label belongs;

Eq(x) = V[x];

With this procedure, we have obtained a matrix containing the labels of all the pixels according to their types of equivalence. Several labels belonging to the same type make the matrix normalization mandatory, tracing it and assigning its type of equivalence to each element.
This algorithm will also store all the elements belonging to each type, reason why an option would be keeping this matrix in memory. This would generate a great waste of memory, since space for spaces in blank (those pixels with no value and not belonging to any type of equivalence) is also reserved. As counterpart, this solution allows us to access directly the type of equivalence of a certain pixel.

Let p = (x,y) pixel ∈ I image.
Eq(p) = I(x,y);

(We shall recall that images can be viewed as functions taking two whole values x and y, and return the value of the pixel in position (x,y)).
Another solution consists in storing another vector in the vector of the existing types of equivalence, where all the points belonging to that type of equivalence are stored for each position. The definition of type would be:

```
Type
  Point = record
    X,Y : integer // pixels
  End;
  VectorOfPoints = array of point;
  Class = record
    Points: PointVector
    numberOfClass: integer
  end;
```

[4] *q-distance* between two pixels is known as the length of the shortest path linking them.

```
  Classes = array of Class
```

The main advantage of this implementation lies in the memory saving, since only those points belonging to some type of equivalence will be stored; the disadvantage is that finding the class of equivalence of a certain point will imply tracing the vector of types of equivalence, and for each type, tracing the points vector belonging to it. In the best of the cases, this will mean a single access (first class, first point). In the worst: all the points of a given image I belong to some type of equivalence and image I has x pixels of height and y of width in a total of z = x*y. In the worst of the cases, it shall be considered that the searched point is found in the last position of the vector and in the last type of equivalence, and the access to this point will take z accesses to memory.
This worst case is not a hundred percent real, since it is impossible that all the points of an image belong to some type of equivalence, since this would mean that it has, for instance, all of its pixels black. From the experience acquired, it is possible to estimate (for the type of images we have worked with) that less than the half of the pixels are black and, in some cases, the number only reaches the 20%.

4. A ZHANG-SUEN'S THINNING ALGORITHM.

This method is fast and simple to implement; it counts with two subiterations in which those pixels fulfilling the rules defined for iteration are removed. Recalling that:

- Label 6 belongs to the type of equivalence 4.
- A pixel is a final point if it has a single black neighbor, being the rest all white.
- A pixel's **connectivity** is defined as the number of objects it could connect in the original image, and is computed turning round a pixel clockwise and counting how many color changes are produced. The number of changes will be the connectivity, i.e., the number of regions it connects.

As a first step of the algorithm, an image smoothing is applied, and all the pixels with two or more black neighbors and with connectivity inferior to two[5] are deleted. Then, the two iterations are carried out. In order to remove a pixel in the first iteration, it should have the following properties:

- Count with connectivity 1.
- Have a quantity of black neighbors between 2 and 6 (included).
- Have at least one of the following pixels in white: [x-1,y], [x,y+1], [x,y-1]
- Have at least one of the following pixels in white [x-1,y], [x+1,y], [x,y-1]

In the second iteration, the pixels fulfilling the following rules will be removed:

- Count with connectivity 1.
- Have a quantity of black neighbors between 2 and 6 (included).
- Have at least one of the following pixels in white: [x-1,y], [x,y+1], [x+1,y]
- Have at least one of the following pixels in white: [x,y+1], [x+1,y], [x,y-1]

As previously expressed, iterations will go on as points are being eliminated. If the pre-processing of the image is not carried out, the results are not so good; results are shown in Figure 1 and 2.

[5]Note that this requires an extra processing, reducing the algorithm efficiency.

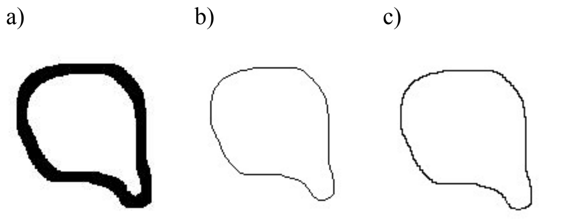

Figure 1: a) Original Image b) Skeleton of the image obtained with the algorithm based on Connected Components c) Skeleton obtained with the Zhang-Suen method.

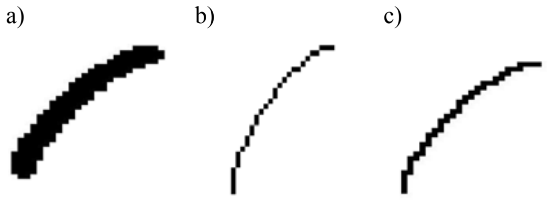

Figure 2: Similar to Figure 1) Here we can notice in detail how the lines of each method are drawn; a greater thickness is observed in the case of the Zhang-Suen method.

5. PROBLEM IN NOISY BORDERS.

According to the experiments carried out, the results have not been so encouraging in some images, for instance, in those cases in which the borders were dithering or have random noise, case in which, the thinning process produces short random lines, generating a distortion of the image or a loss of its shape. The problems presented in this type of images lead to drawbacks when characters are to be recognized.
Those images presenting this characteristic have a "dirty" border with points distributed at random and, when they are thinned, the random points are not removed since they fulfill the characteristics of the border point. Both algorithms react in the same fashion before this phenomenon, generating lines all along the border of the image, similar to ramifications of its skeleton, in an attempt to keep the border points together with such skeleton. (See Figure 3).

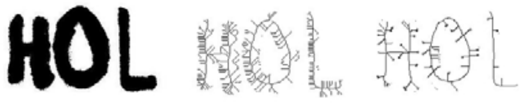

Figure 3. Skeletons of the first image obtained with CC in the first case and with Zhang-Suen in the second.

A solution with filters. This problem can be solved by applying a mean filter to the image before it is thinned, thus producing smoothest borders and without the previous imperfections. In the results obtained (see Figure 4), we have attained clearly better image skeletons, especially with the Zhang Suen algorithm, since with the CC-based method we not always obtain the same improvements. As disadvantage, this method adds an extra processing cost.

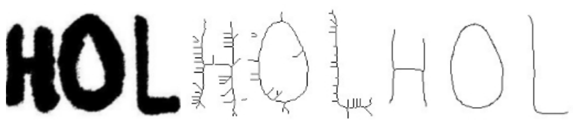
Figure 4. Image to which the mean filter was applied; Skeleton of the same image with CC; Skeleton of such image with Zhang-Suen.

6. OPTIMIZATION WITH THE CONNECTED COMPONENTS – BASED METHOD.

The problem of the thinning algorithm based on the computation of connected components is its high complexity and inefficiency. It can be upgraded by using another method for the recognition of simple points (main use of the algorithm) without the need of obtaining all the connected components.

This technique has the same basis as the connected components-based method: the removable points are those which are simple and are not final point, though it requires considerable lesser time to determine whether a point is simple. The idea of this optimization consists in verifying each point in a given side; this means that a point may be viewed as a simple point in a direction if its neighbors fulfill certain conditions **in such direction.**

The algorithm carries out several runs, verifying in each a different size: north, south, east, and west sides. In each of the runs, the image is traced, searching those black points which are simple according to the sense of the tracing and which are not a final point.

Let d be the direction in which point P is being analyzed. Let P_d be the set of conditions indicated if point P is simple in direction d. Of course, for each possible d (north, south, east, and west), the set P_d will be different. The elements of the set P_d are called c_i functions

c_i : (Image, Point) -> [F,T] (F = FALSE, T = TRUE)

The structure of the set Pd is similar for all of the cases: there exists a set of conditions that should NOT be fulfilled and others that should be necessarily fulfilled. A common condition is that point P should have the black value, which will be here ignored as member of the set (since black points are always being analyzed).

Let I(x,y) : Image -> [0,1] be *function returning the value of point (x,y). (1= white, 0 = black)*

Let Sn:: Point -> [F,T] Sn be the function indicating whether a point if north simple. (F = FALSE, T = TRUE)

d = North (1)

$$Sn(p)\begin{cases} T \text{ if } I(x-1,y)=1 \wedge \neg\exists\, c_i \in P_d \\ \qquad c_i = T \text{ with } i=1,2,3,4,5 \\ F \text{ otherwise} \end{cases}$$

- c1= I(x,y-1) =0 ^ I(x+1,y) =1 ^ I(x,y+1) =0
- c2= I(x,y-1) =1 ^ I(x-1,y-1) =0 ^ I(x-1,y) =1
- c3= I(x-1,y) =1 ^ I(x-1,y+1)=0 ^ I(x,y+1) =1
- c4= I(x,y+1)=1^ I(x+1,y+1)=0 ^ I(x+1,y) =1
- c5= I(x+1,y)=1^ I(x+1,y-1) =0 ^ I(x,y-1) =1

Figure 5 graphically shows the 5 conditions.

Black and white points are those points that should have that value. Gray points are points whose value is of no interest. The point marked with an (x) is the analyzed point.

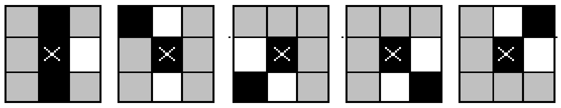
Figure 5: Cases allowing the identification of a north simple point.

Similarly, Ss(p) (South Simple Point), Se(p)(EastSimple Point) So(p) (West Simple Point) are defined as:

d = South (2)

$$Ss(p)\begin{cases} T \text{ if } I(x+1,y)=1 \wedge \neg\exists\, c_i \in P_d \\ \qquad c_i = T \text{ with } i=1,2,3,4,5 \\ F \text{ otherwise} \end{cases}$$

- c1 = I(x-1,y)=1 ^ I(x,y+1)=0
- c2, c3, c4 and c5 are the same

d = Est (3)

$$Se(p)\begin{cases} T \text{ if } I(x,y+1)=1 \wedge \neg\exists\, c_i \in P_d \\ \qquad c_i = T \text{ with } i=1,2,3,4,5 \\ F \text{ otherwise} \end{cases}$$

- c1 = I(x-1,y-1)=1 ^ I(x-1,y)=0 ^ I(x+1,y)=0
- c2, c3, c4 and c5 are the same

d = West (4)

$$So(p)\begin{cases} T \text{ if } I(x,y-1)=1 \wedge \neg\exists\, c_i \in P_d \\ \qquad c_i = T \text{ with } i=1,2,3,4,5 \\ F \text{ otherwise} \end{cases}$$

- c1 = I(x,y+1)=1 ^ I(x-1,y)=0 ^ I(x+1,y)=0
- c2, c3, c4 and c5 are the same

As observed, conditions c2,c3,c4 and c5 are always the same and have been included in the set P_d for the sake of the algorithm's implementation, in which an *or logic* is carried out among all the conditions of the set (then, such result is rejected in order to represent the $\neg\exists$ of each function's definition).

Once the points fulfilling the simple, and not the final, point conditions (in the corresponding direction) are found, they are removed changing their values into white. This will be repeated, like in the previous algorithm, while points fulfilling some conditions are being found.

7. DIFFERENT BORDERS.

Figure 6 shows the images, two of which have been thinned with each of the described algorithms; the thinning in both cases (b and c) is right, keeping the topology and without unexpected effects, but if we observe in more detail, we could see that the line generated by the CC algorithm is thinner than the line generated by Zhang-Suen.

Figure 6 shows three images. The first is a line thinned using CC, and the third is the same line, though Zhang-Suen was applied. The central image shows in gray the new points emerging when using the latter algorithm.

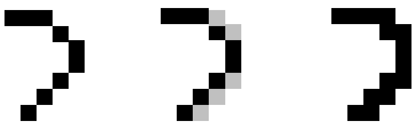

Figure 6: Lines 1 and 3 have been thinned with CC and Zhang-Suen methods, respectively. The central line shows the "difference" between both.

This is due the fact that, by using 4-adjacency in CC, in order to keep a line linked (within the same CC), only its vertices are to be handled. While in this case the problem does not get worse, in several instances it is desirable to count with more defined lines. The solution to this particular situation is quite simple: instead of taking 4-adjacency, use 8-adjacency when finding the CC; i.e., instead of considering just two points, 4 points are to be considered. Figure 7 shows in gray, in the first place, the points considered with 4-adjancency, while next they are presented with 8-adjacency.

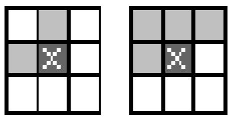

Figure 7: Pixels to be analyzed if we take 4-adjacency and 8-adjacency.

When using 8-adjacency, we get similar results in both algorithms.

8. CONCLUSIONS.

Even though both algorithms render proper results, the Zhang-Suen's algorithm proved to be simpler to implement and, in terms of empirical efficiency, we observed that this algorithm typically demands inferior running times than the CC-based algorithm, even in its improved version. Data can be seen in Figures 8 and 9, where the times spent in the running of each algorithm in its simple version, and after applying filters, are graphically shown.

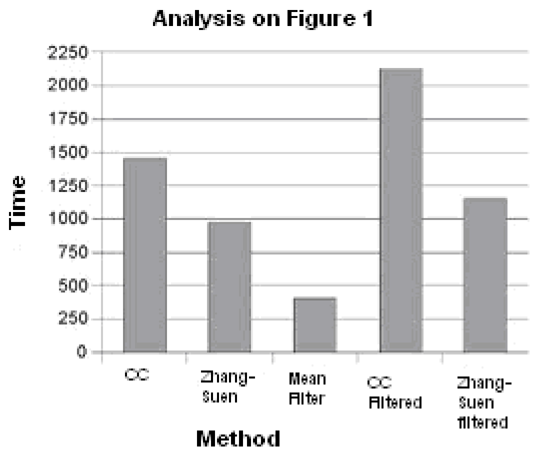

Figure 8 Analysis of the times for the different techniques developed over the image of Figure 1.

Another advantage of the Zhang-Suen's algorithm is its treatment of borders with random noise. As previously mentioned, when a mean filter is applied over the image, we can obtain satisfactory results, while the CC-based algorithm not always presents those results.

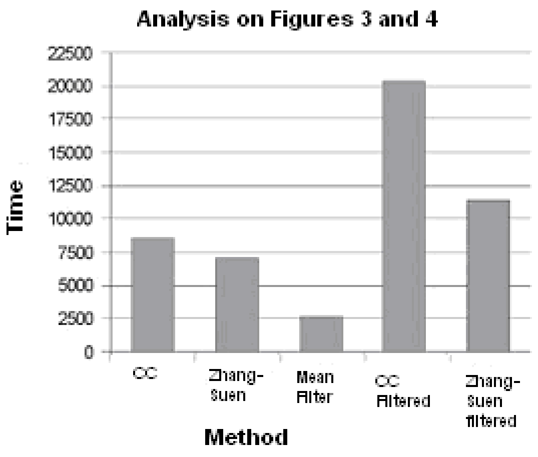

Figure 9: Time Scheme applied to Figure 3 and 4 with the different techniques

However, this does not mean that it is always better to use the former algorithm. One of the problems found with the Zhang-Suen's algorithm is that if there exist intersected thick lines, when they are thinned, they generate an effect in which such intersection produces a segment instead of a simple point. This is known as "*necking"* effect; and in order to solve it the image should be pre-processed, focusing on its close angles. But this entails more processing, which is translated in extra running time. The "necking" effect, though it exists, is not so marked when using CC.

On the other hand, an image connected components are also used for other purposes; for instance, to know the quantity of closed areas in an image (holes), reversing the image (changing the whites to blacks and blacks to whites). Then, all the points belonging to the same CC will belong to the same closed area; and the total quantity of closed areas will be the quantity of connected components minus 1. The method is also used as basis for segmenting the image in words, taking in principle elements of the same connected component as simple elements.

If we take a global look to the system, using connected components may benefit the efficiency, since they are computed only once and they are then used as many times as is required.

REFERENCES.

[1] Elisabetta Bruzzone, Meri Cristina Coffetti; Elsag spa – R&P Departament. An algorithm for Extracting Cursive Text Lines, Genova – Italy

[2] Radmilo M. Bozinovic, Sargur N. Srihari; "Off-Line Cursive Scrtipt Word Recognition". IEEE Transactions on Pattern Analysis and Machine Intelligence. Vol 11 No. 1

[3] K. Badie and M. Shimura. Machine recognition of roman cursive script in Proc. 6th. Int. Conf. Patternt Recognition, Munich, West Germany, Oct 1982.

[4] R. Manmatha, Chengfeng Han, E. M. Riseman and W. B. Croft; "Indexing Handwriting Using Word Matching". Center for Intelligent Information Retrieval, Computer Science Department, Univeristy of Massachusetts, Amherst.

[5] Toni M. Rath and R. Manmatha; Features for Word

Spotting in Historical Manuscipts. Multi-Media Indexing and Retrieval Group. Center for Information Retrieval. University of Massachusetts, Amherst.
[6] L. Fletcher and R. Kasturi. “A robust algorithm for text string separation from mixed text/graphics images”. IEEE Transactions on Pattern Analysis and Machine Intelligence, 10:910-918. 1988
[7] 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
[8] D. Wang and S. N. Srihari. “Classification of newspaper image blocks using texture analysis”. Computer Vision Graphics and Image Processing, 47:329-352
[9] Michel Weinfeld, “Reconnaissance d'ecriture manuscrite: segmentation de mots”. Département d'Enseignement et de Recherche en Informatique. École Polytechnique, Paris, France
[10] William Prat, John Wiley & Sons, Digital Image Processing, 1991, Second Edition
[11] Gonzalez Rafael, Woods, Addison-Wesley Digital Image Processing, 1992, Second Edition
[12] T. M. Rath, R. Manmatha. “Word Spotting for Handwritten Historical Document Retrieval”

Received: Jul. 2005. Accepted: Sep. 2005.