1988

Digital character scaling by contour method.

Abderrahmane. Namane
University of Windsor

DIGITAL CHARACTER SCALING BY CONTOUR METHOD

by

Abderrahmane NAMANE

Submitted to the
Faculty of Graduate Studies and Research
through the Department of
Electrical Engineering in Partial Fulfillment
of the requirements for the Degree
of Master of Applied Science at
the University of Windsor

Windsor, Ontario, Canada
1988

ISBN 0-315-48182-X

DEDICATION

TO MY PARENTS

ABSTRACT

An algorithm for digital character scaling by a contour method is developed and implemented.The digitized image of the font to be scaled is obtained by means of a vidicon camera .The character must be thresholded before processing with this method.

The algorithm is based on scaling the contour of the character through a transformation, cubic splines are used to interpolate the discrete samples of the contour character. The algorithm is applied to Arabic characters.

AKNOWLEDGEMENTS

I Would like to express my sincere appreciation to my supervisor, Dr. M. A. Sid-Ahmed for his valuable suggestions, guidance, support, and helpful remarks throughout the course of this work.

I am also most grateful to my committee members; Dr. J. J. Soltis, Dr. H. K. Kwan and Dr. A. Mansour for their constructive suggestions.

I wish to thank all my fellow graduate students and my colleques, M. TELLACHE , M. M. MAMERI , F. EL-KHALI , S. O. BELKASIM. I would like to thank all my friends from Electrical(power) and Civil Engineering departments.

Last but not least, I extend my sincerest thanks to all my beloved parents, my brother AHMED, SALAH and all my family in ALGERIA.

TABLE OF CONTENTS

FIGURE_CAPTIONS

Chapter I
INTRODUCTION

The advancement in the digital image processing hardware has provided the printing industry with new facilities for capturing new fonts.A font is a group of character types of one style and size.

The importance of enlarging and reducing two level images such as characters in typesetting and graphical text, continues to grow as more such characters are digitally represented. Digital character scaling is the process performed on the digital character input ,resulting in a digital character output of a different size. For this purpose a binary image is defined as a two-dimensional signal whose amplitude is precisely either black (numerically and logically represented as 1) or white (numerically and logically represented as 0).

Very little work has been presented in the literature for scaling binary images of fonts.The work done by R. Ulicheny and D. Troxel [1],[2] utilizes telescoping template.The templates could be of any size, and the quality of the scaled font is dependent on the size of the template.The larger the size , the better the quality. Results for up to third-order window scaling are presented in this work.These results are reasonably good except that for very large enlargement there is an appearance of jaggies.

A well known method for font scaling is that of replication [3].This however,resulted in pronounced jagged edges when the character was magnified.

Although few papers have been presented for the scaling of fonts represented as binary images, for multilevel images a large number of algorithms have been investigated(eg. [4]-[6]).These algorithms are based on interpolation techniques ,and are aimed at increasing the dimensionality of the whole image rather than a particular object.These algorithms do not lend themselves to scaling of binary images, since the scaled image will have to remain binary.

Knuth [7],used mathematical approaches such as spline curves, circles and straight lines for font design.

The importance of printing in advertising ,marketing and sales literature, text books...etc,shows the role that fonts play. Fonts are useful in differentiating headings, paragraph titles , and photo captions, for example , by styles ,sizes, weights, and emphasis. Typeset material is easier to read and has an effect on both eyestrain and reading interest in that good fonts improve perceived print quality.Type sizes are measured in units called "points", a system of measurment used exclusively in typography. The typographic point is approximately 1/72 of an inch.

To add flexibility to a word processor,algorithms should be included that can operate on camera captured fonts rather than a fixed set of designed fonts.

In this thesis a review of the state-of-the-art will be carried out,and a new method for scaling of fonts is presented and compared with the techniques reviewed .

This investigation focuses on an operation on the character size of a given font,resulting in a character of a different size by either magnification or minification using the border points of the font in a scaling mapping and smoothing algorithms.

1.1 GOAL OF THIS THESIS :

This thesis develops :

1.1.1 The use of contours in character scaling :

The images are camera captured fonts. In this case Arabic characters are used.

The camera captured font is processed by many operations such as :

1) Thresholding
2) Border detection
3) Border scaling
4) border interpolation
5) Filling in between the contours

Using a thresholding technique [8] in which all gray levels below the threshold value are mapped black and those levels above are mapped white, the results in a binary image. A border following algorithm [9] is used to extract the contours. By using a scaling transformation,all pixels belonging to the font are mapped outward or inward depending on the scaling ,either magnification or minification . Resulting discrete samples (in the case of magnification) are interpolated by mean of cubic splines [10]. Finally, filling in between the border is performed [11], which results in the desired scaled character. Also, generation of Arabic fonts by means of spline curves with the minimum number of points is carried out.

1.2 Thesis organisation

Chapter II gives a brief account of multilevel image scaling. Several of the techniques available in the literature, have been presented. A brief description of why those techniques are not practical in scaling of binary images, which are fonts in this case is given. Then follows a detailed explanation of some of the scaling of binary images techniques. Results are shown for those techniques, and a discussion is developed.

Chapter III gives a detailed discussion of scaling by the contour method. Examples are given to illustrate the use of the contour method. Results are shown and a discussion is developed. As well, an explanation of how a font is generated by means of computer is given, and this procedure is applied to the Arabic character.

Chapter VI finally develops a general comparison and discusses the derived conclusion.

Chapter II

SCALING_OF_MULTILEVEL_AND_BINARY_IMAGES

2.1 _SCALING_OF_MULTILEVEL_IMAGES

2.1.1 _INTRODUCTION

In relation to the many applications of interpolation in signal processing (see [12]), the need for a sampling rate constantly arises in image processing. Examples of such applications are image resolution conversion and image change of scale. The process of decreasing the data rate is called decimation, and increasing data samples is termed interpolation.The resolution conversion process can be seen as a two-step operation.First,the discrete data is reconstructed (interpolated) into a continuous curve,then it is sampled at a different sampling rate,as shown in Fig.(2.1).In real digital processing,the procedure of reconstruction by interpolation and sampling at a different rate can be done in one operation.

In this section two types of interpolation are presented. One is based on interpolation with a one-dimensional formula applied to every row then to every column of the image to be interpolated, the other is based on interpolating by surface over a given rectangle region, the assumption being that the image to be interpolated is a concatenation of a finite number of rectangle regions.

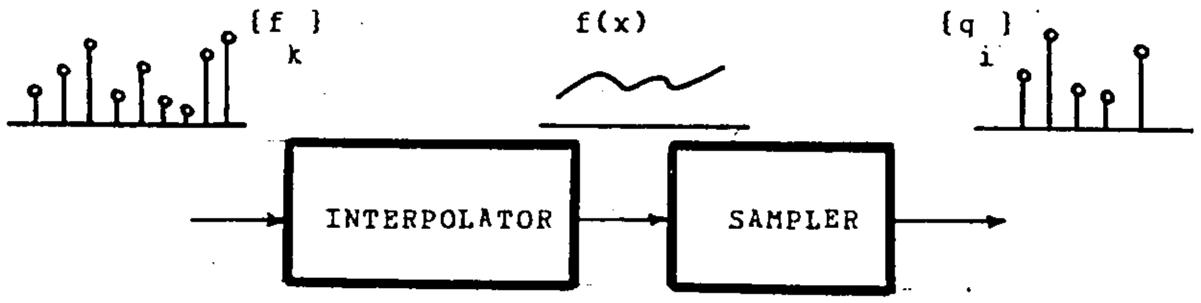


Figure_2.1: The resolution conversion process.

2.1.2 CUBIC_SPLINES_FOR_IMAGE_INTERPOLATION_[5]

In this type of interpolation [5], a one-dimensional interpolation formula must be evaluated (see chapter VI).

$$f(x)=\sum_{k=1}^{K} C_k S_k(x) \qquad (2.1)$$

where C_K are the coefficients to be determined from the input data, $S_k(x)$ are chosen basis functions, and K is the number of given data points.

Having found the coefficients C_K from the input data, the equation (2.1) is applied to every row then to every column.The two-dimensional interpolation requires (m+N) one-dimensional interpolations to be executed ,if m is the number of rows of data points and N is the size of the output image.

2.1.3 INTERPOLATION OF DIGITAL IMAGERY USING HYPERSURFACE APPROXIMATION [6]

The hypersurface approximation [6] is carried out by a quadratic surface, defined over a two-dimensional space of the digital picture in the neighbourhood of the point to be interpolated, using orthogonal polynomials as basis functions. Given f(X); digital picture function, an estimate of f(X) is given by :

$$g(X)=\sum_{i=0}^{N} a_i S_i(X) \qquad (2.2)$$

Where $X=(x_1,x_2)$ a point in the two-dimensional space,and $(a_i, 0<i<N)$ are sets of coefficients and $(S_i(X), 0<i<N)$ a set of two dimensional orthogonal basis functions.

The total squared estimation error, E^2, can be written as:

$$E^2 = \sum_{x \in r0} (f(X)-g(X))^2 \qquad (2.3)$$

Using the orthogonal properties of the basis functions, from equation (2.2) and equation (2.3), the coefficient a_i that minimizes E^2 can be obtained as ([4]):

$$a_i = \sum_{X \in r0} f(X) \cdot S_i(X) / \sum S_i^2(X) \qquad (2.4)$$

Having found the coefficients a_i and $s_i(X)$, equation (2.2) is applied to every rectangle region to be interpolated.

2.2 REPLICATION_AND_TELESCOPING_TEMPLATE_METHODS_FOR_FONT SCALING

2.2.1 _INTRODUCTION

The algorithms discussed above are based on interpolation techniques, and they are aimed at increasing the dimensionality of the whole image rather than a particular object. These algorithms do not readily lend themselves to scaling of binary images, since the scaled image will not necessarily remain binary. For this purpose, two methods of scaling of binary images are presented next.

2.2.2 REPLICATION_[3]

The replication method consists of repeating each pixel belonging to the object inside mxm square ,where m is the linear magnification factor (see Fig.(2.2)). In other words, the interpolating basis function is the sample-hold function B0 , as shown in Fig.(2.3).

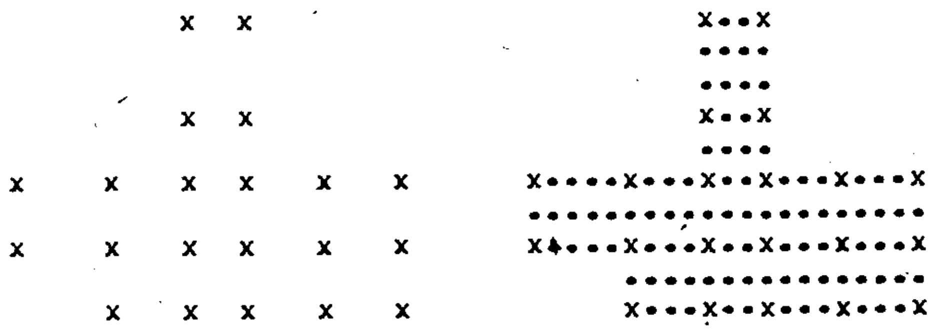
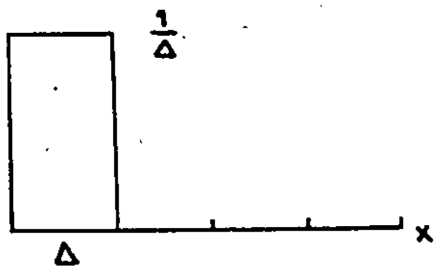
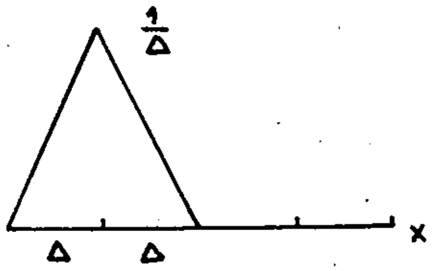


Figure 2.2: (x) represents the original pixel, and (.) the added pixel.



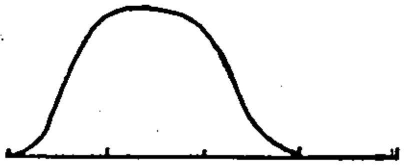
$B0(x;\ x_k, x_{k+1})$

Smaple-hold function . ($\Delta = x_K - x_{K-1}$)



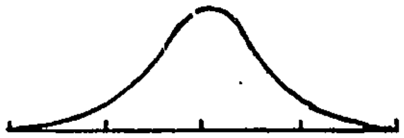
$B1(x;\ x_{k-1}, x_k, x_{k+1})$

Chateau function .



B2(x; x_{k-1}, x_k, x_{k-1}, x_{k+1})

Quadratic.



B3(x; x_{k-2}, x_{k-1}, x_k, x_{k+1}, x_{k+2})

Cubic .

Figure 2.3: SKETCH OF THE FIRST FOUR LOWER ORDER B-SPLINES.

2.2.3 TELESCOPING TEMPLATE METHOD [1-2]

Ulichney and Troxel [1] ,presented the telescoping template method.In this method the contour characteristic that can occur within a given image "window" is stored in a telescoping template, which consists of a concatenation of many unit squares. Fig.(2.4) illustrates an assignment area with its associate first,second and third order window.

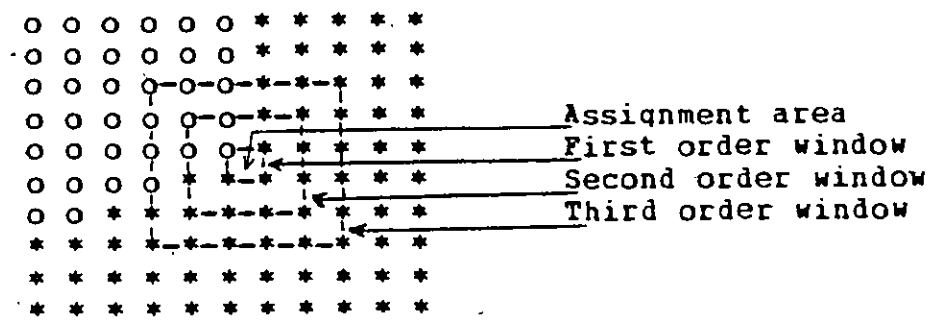


Figure 2.4: ASSIGNMENT AREA AND SOME POSSIBLE WINDOWS.

Once an assignment rule is selected for each assignment area describing how each assignment area is painted then the reconstruction is complete. This is based on the neighborhood of samples defined by a window center.

If the window order p is small, direct enumeration of all $2^{(2p)^2}$ window arrangements would be tedious but nevertheless manageable.

The telescoping template used by [1] is illustrated by Fig.(2.5), where it shows one particular assignment area and the appropriate assignment rule for windows of increasing order.

Window	scaling order	interpretation	assignment rule
*	p=0	Solid black	■
o * * *	p=1	45 angle	◢
o o * * o o * * * * * * * * * *	p=2	90 inside corner	□
o o o * * * o o o * * * o o o * * * o * * * * * * * * * * * * * * * * *	p=3	120 inside corner	◪

Figure 2.5: TELESCOPING TEMPLATES.

Window decoding is simple and has two steps.In the first step ,the first order window is considered,only sixteen possible arrangements exists and fall into two groups,enumerated here with their associated assignment rules (see Fig.(2.6)).

(1) Solid area group.

$\begin{matrix}0&0\\0&0\end{matrix}$	$\begin{matrix}0&1\\1&0\end{matrix}$	$\begin{matrix}1&0\\0&1\end{matrix}$		□
$\begin{matrix}1&1\\1&1\end{matrix}$				■

2) Edge area group.

$\begin{matrix}0&0\\0&1\end{matrix}$	$\begin{matrix}0&1\\0&0\end{matrix}$	$\begin{matrix}1&0\\0&0\end{matrix}$	$\begin{matrix}0&0\\0&0\end{matrix}$	□
$\begin{matrix}0&0\\1&1\end{matrix}$	$\begin{matrix}0&1\\0&1\end{matrix}$	$\begin{matrix}1&1\\0&0\end{matrix}$	$\begin{matrix}1&0\\1&0\end{matrix}$	□
$\begin{matrix}0&1\\1&1\end{matrix}$	$\begin{matrix}1&1\\0&1\end{matrix}$	$\begin{matrix}1&1\\1&0\end{matrix}$	$\begin{matrix}1&0\\1&1\end{matrix}$	◪

<u>Figure 2.6</u>: SIXTEEN POSSIBLE ARRANGEMENTS AND THEIR ASSOCIATED ASSIGNMENT RULES.

Note that for the second order window, thirteen templates and eight assignment rules were presented, for the third window forty five templates and twenty two assignment rules are used. A complete list of templates and assignment rules is given [2], with corresponding assignment rules.

2.3 RESULTS

The computer simulation results obtained by using the above two methods are shown in Figs.(2.7)-(2.16). Among the images in Figs.(2.7)-(2.10) are those obtained from the replication, and the images in Figs.(2.11)-(2.16) are those obtained by the telescoping template method of window order 1, 2 and 3.

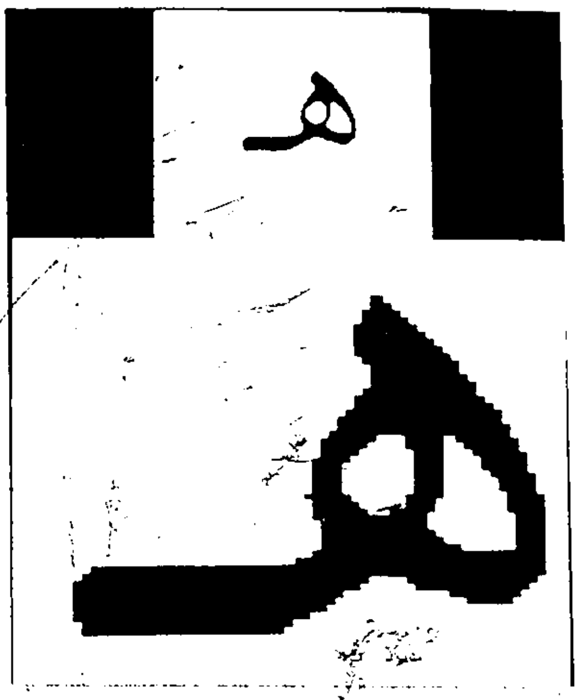


Figure 2.7: Replication method for Arabic character, scalx=4.0 and scaly=4.0, character with three contours.

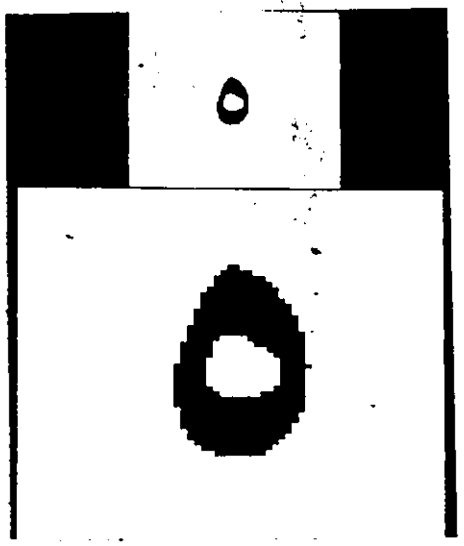


Figure 2.8: Replication method for Arabic character, scalx=4.0 and scaly=4.0, character with two contours.

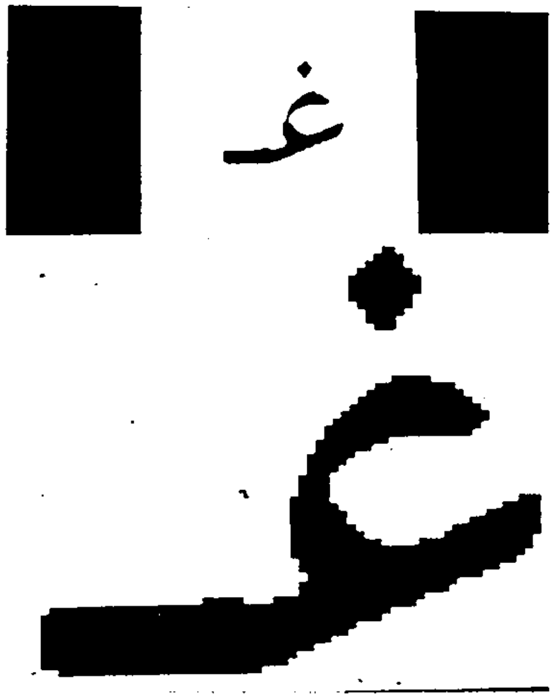


Figure 2.9: Replication method for Arabic character, scalx=4.0 and scaly=4.0, character with dot.

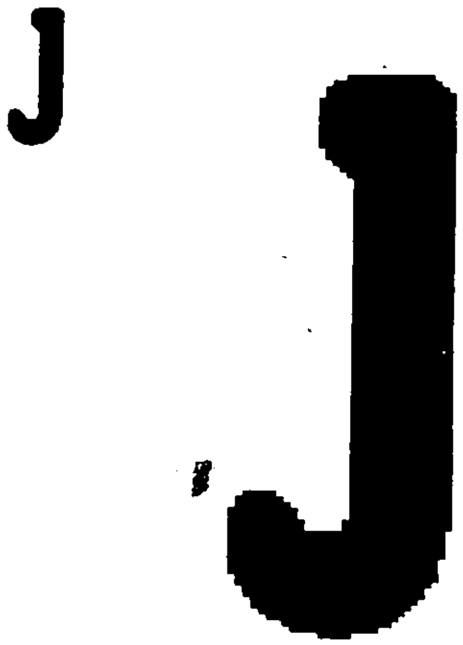


Figure 2.10: Replication method for English character, scalx=4.0 and scaly=4.0 , character with one contour.

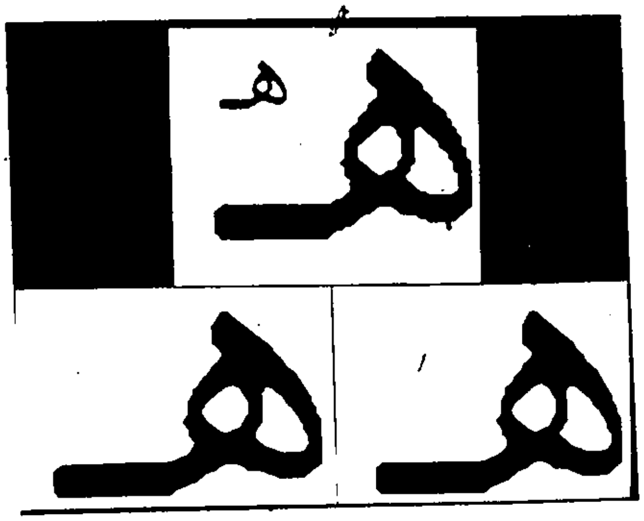


Figure 2.11: Telescoping template method for Arabic character, scalx=4.0 and scaly=4.0, character with three contours; a) given character, b) First order c) Second order, d) Third order.

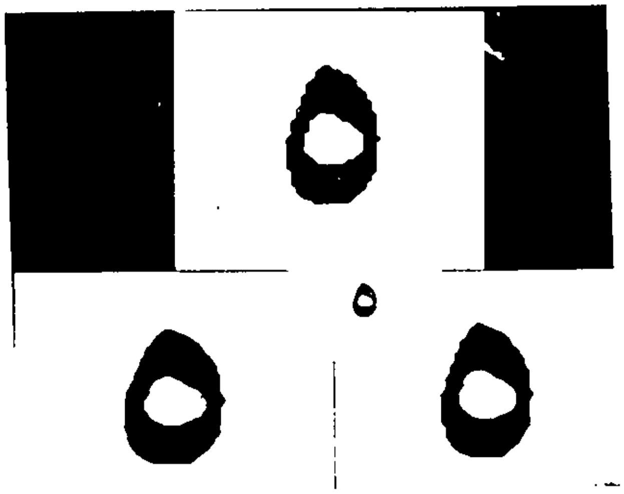


Figure 2.12: Telescoping template method for Arabic character, scalx=4.0 and scaly=4.0, character with two contours; a) given character, b) First order, c) Second order, d) Third order.

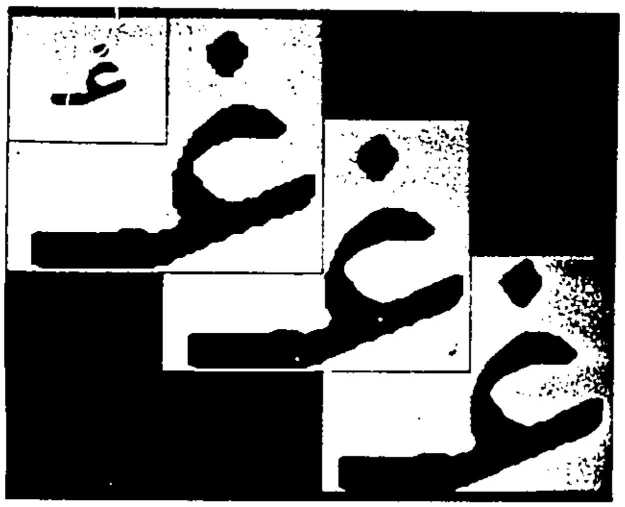


Figure 2.13: Telescoping template method for Arabic character, scalx=4.0 and scaly=4.0, character with dot; a) given character, b) First order, c) Second order, d) Third order.

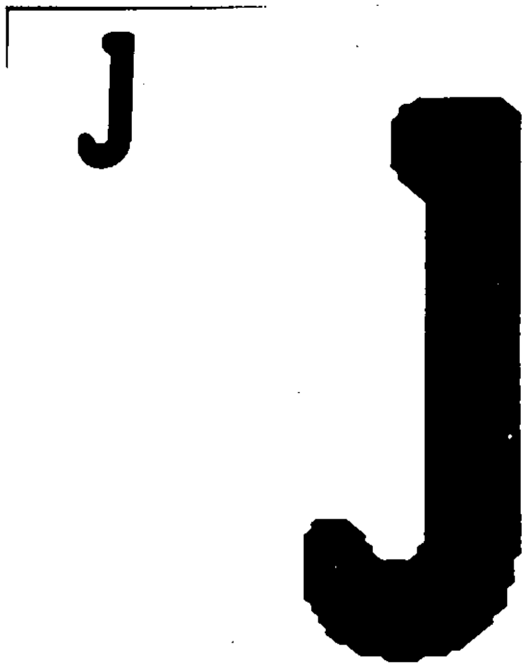


Figure 2.14: Telescoping template method for English character, using first order window, scalx=4.0 and scaly=4.0 .

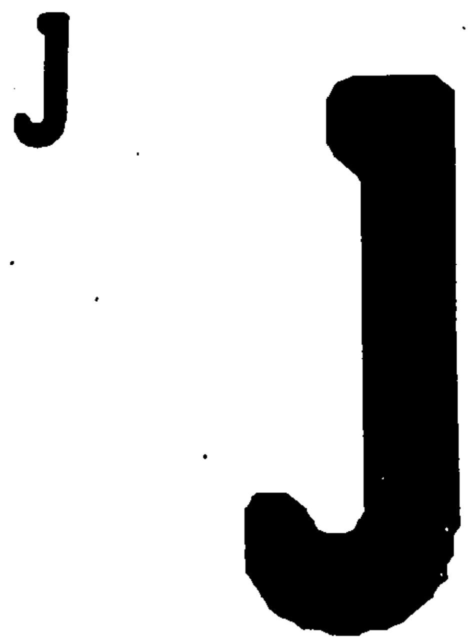


Figure_2.15: Telescoping template method for English character, using second order window, scalx=4.0 and scaly=4.0 .

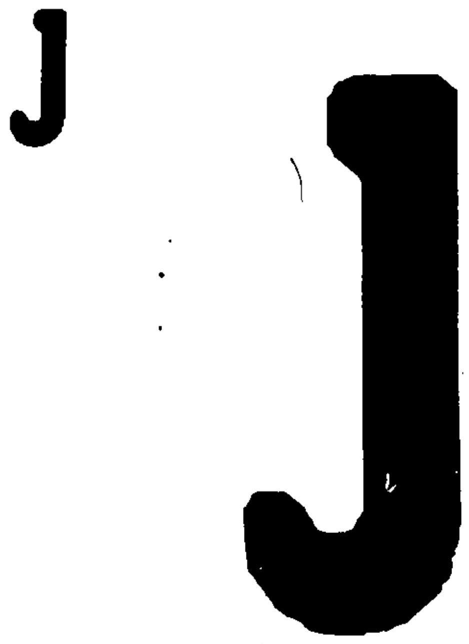


Figure_2.16: Telescoping template method for English character, using third order window, scalx=4.0 and scaly=4.0 .

2.4 CONCLUSION

The replication has resulted in highly pronounced "jaggies" along the edges, the greater the enlargement the higher the jaggies. In the telescoping template method , the characters still have jagged edges and the time requirement increases with the window order. As can be noticed the quality of the scaled font is dependent on the size of template , the larger the size the better the quality. Results provided by this method are good except when great enlargement occurs and jaggies appear. In conclusion the only problem faced in character scaling is the stairy shape of the edges.

Chapter III

A_NEW_SCALING_ALGORITHM

3.1 _INTRODUCTION

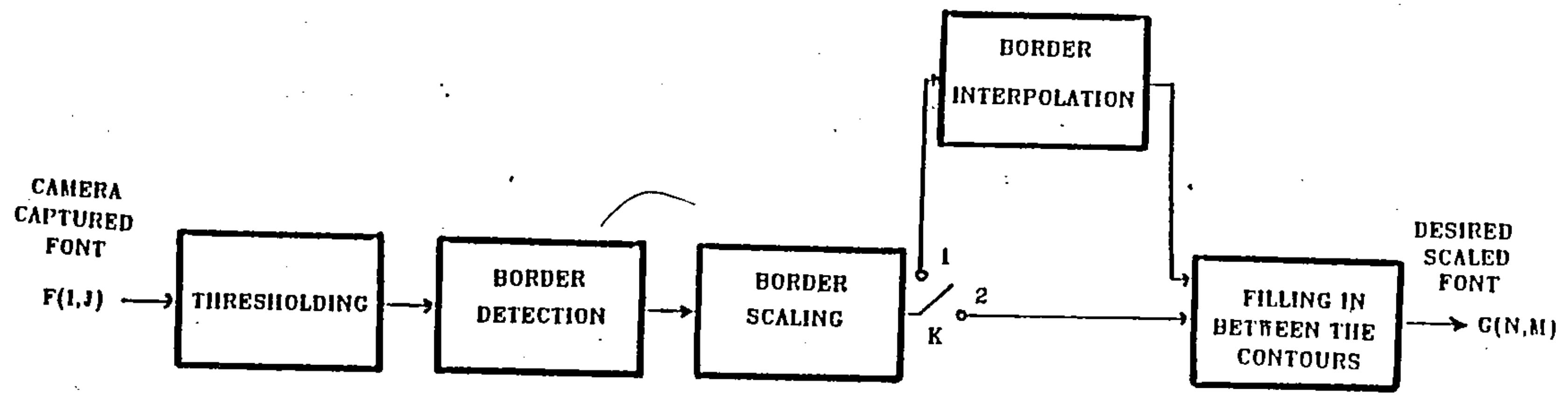
A new scaling algorithm is developed to investigate the problem discussed above . This problem occurs only within the border. Thus the role of this algorithm is to work on that border (contour). The contours are scaled, interpolated to give very smooth edges and meanwhile eliminate jaggies. The block diagram shown in Fig.(3.1) is described next.

3.2 GRAY_LEVEL_THRESHOLDING.

The goal of thresholding is to partition a given image into meaningful regions,by transforming the continuous tone image (gray-levels 0,1,2,.....,255) to a binary or multilevel image,and at the same time retain all necessary features of the original image.

This operation consists of dividing the gray level scale into bands, and then using thresholds to determine regions or to obtain boundary points.However in most of the cases not all the levels are fully utilized in defining the image. This will be more obvious when the probabilty distribution of the gray levels is plotted. The plot of the probability distribution is often termed as the histogram of the image. By looking at the histogram of an image it is

BLOCK DIAGRAM



K ON 1 ; MAGNIFICATION

K ON 2 ; MINIFICATION

Figure_3.1: CONTOUR METHOD BLOCK DIAGRAM .

seen that more often they are clustered into two distinct groups as shown in Fig.(3.2). These groups normally represent the two populations of the image ;the object and the background, all pixels with gray level value below the threshold value T are mapped black and those gray levels above are mapped into white (this applies to the character in this case). This technique is called single-level thresholding and T is Known as the threshold value. However there might be cases where the image might contain more than two distinct populations, as shown in Fig.(3.3).In these cases it is required to group the levels into more than two values.This type of segmentation of a given image into different regions is called multi-level thresholding with threshold values T1 and T2.

Threshold selection for images whose histograms are the same shape as the one given in Fig.(3.2) or(3.3) is quite straightforward.The threshold value is selected at the bottom of the valley between two peaks. Methods exists which consist of transforming the histogram of an image to a shape where threshold selection would be easier.Such techniques have been investigated by several authors [13-15].

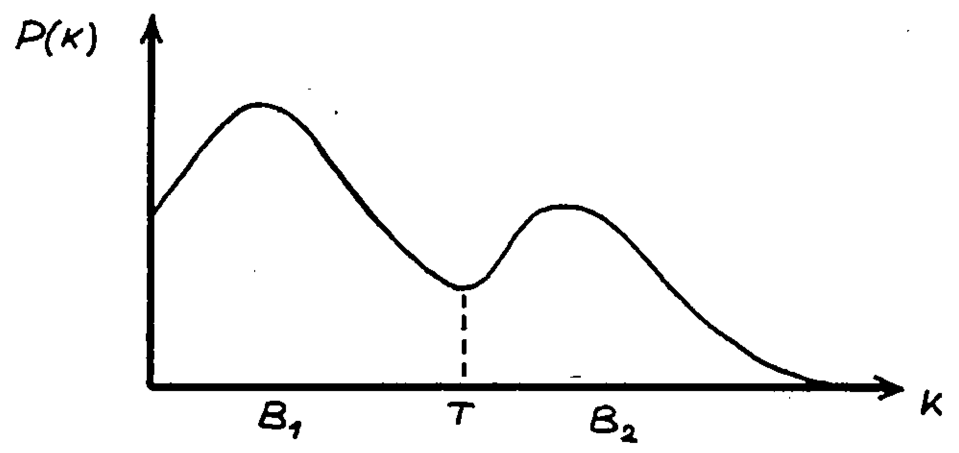


Figure 3.2: A SAMPLE HISTOGRAM ILLUSTRATING A BI-MODAL DISTRIBUTION.

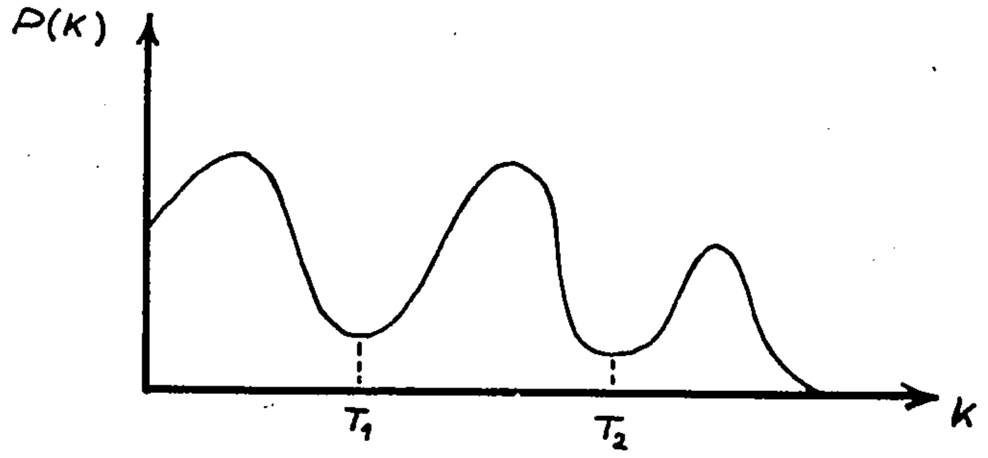


Figure 3.3: A SAMPLE HISTOGRAM ILLUSTRATING A MULTI-MODAL DISTRIBUTION.

3.3 BORDER DETECTION :

The resulting image from the previous block is thresholded using a thresholding value. The two levels are identified as 0 and 1. The border following can then be applied as explained in [9].

The algorithm has been developed in such a manner that it can detect transitions either from 0-1 or from 1-0. However the algorithm interprets in such a way that both the transitions appear to be from 1-0.

In this algorithm, as it can be seen, the border point once detected results in the tracing of the entire border of the region. Hence the border points of any region can be easily stored in array to be used later (i.e, in the contour interpolation). Also, the method of computing the neighbouring point co-ordinates makes the algorithm computationally efficient in detecting borders. The algorithm is tested out on several thresholded images. As an example, consider the thresholded character of the character that was shown in Fig.(3.4b). It is seen that the borders are properly identified in comparing them with the original image (Fig.(3.4a).

Once the borders are extracted , each border is given a separate label, and the number of points on each border is counted. This constitutes a data-base which is utilized in border scaling.For this purpose the border following the algorithm described in [9] is utilized.

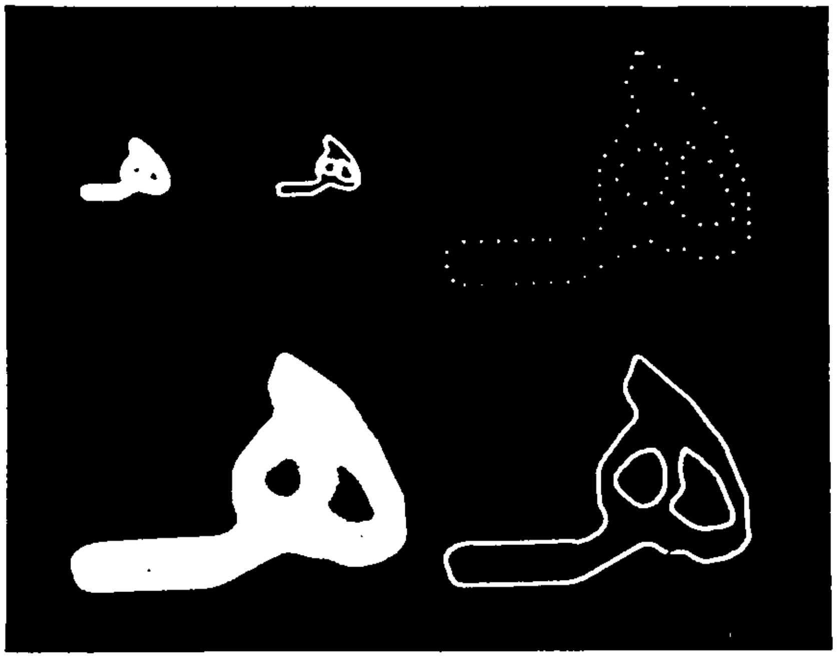


Figure 3.4: Contour method illustration; a) given binary image b) border detection c) border scaling d) border interpolation e) filling in between the borders.

3.4 BORDER SCALING:

In this process each pixel on the border is moved outward or inward based on the following procedure:

1- An object center (see Fig.(3.5)) is calculated as follows:

$$N = \frac{top + bottom}{2} \qquad (3.1)$$

$$M = \frac{left + right}{2}$$

This step is required to avoid the translation of the object to one side of the captured image.

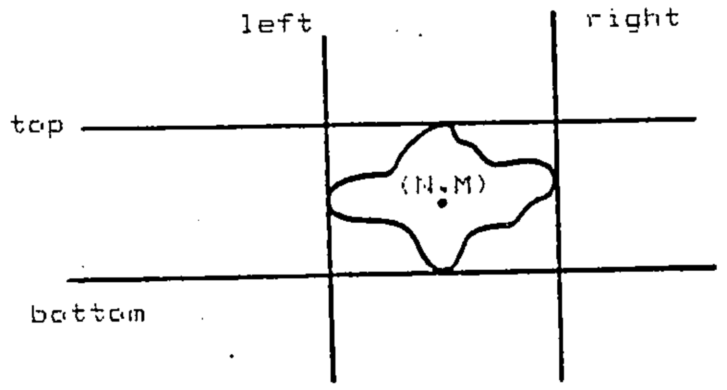


Figure 3.5: Object center.

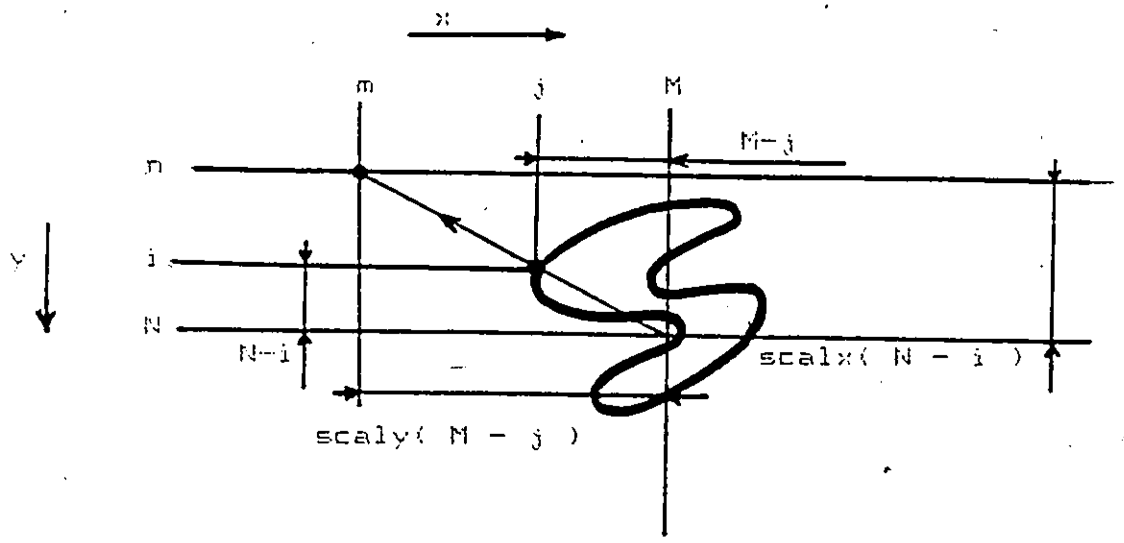


Figure 3.6: Scaling mapping.

2- A scaling transformation is applied :

In this step a transformation to map each pixel belonging to the font, to its new scaled position is obtained. This operation depends on the desired size of the output (scaling factors), and can be done as follow :

Given an image of a font (see Fig.(3.6)), with its center at (N,M) and a point of coordinate (i,j) belonging to the border of the font, it is required to find the transformation which maps :

$$(i,j) \xrightarrow{\;\;H\;\;} (n,m)$$

From Fig.(3.6) we can write :

$$m = M - scaly*(M - J)$$

or :

$$m = scaly*j + M*(1 - scaly)$$

and similarly :

$$n = scalx*i + N*(1 - scalx)$$

The set of the above equation a can be re-written as:

$$\begin{bmatrix} n \\ m \end{bmatrix} = \begin{bmatrix} scalx & 0 \\ 0 & scaly \end{bmatrix} \begin{bmatrix} i \\ j \end{bmatrix} + \begin{bmatrix} N * (1 - scalx) \\ m * (1 - scaly) \end{bmatrix} \qquad (3.2)$$

Where scalx and scaly are the scaling factors over x and y respectively. As an illustration see Fig.(3.7).

3.5 INTERPOLATION :

3.5.1 INTRODUCTION :

Interpolation is the process of estimating the intermediate values of a continuous event from discrete samples. Interpolation is used extensively in digital image processing to magnify or reduce images. In principle we are seeking a smooth continuous curve passing through a set of discrete data at certain spatial points. Mathematically speaking, the interpolated continuous function in one dimension is :

$$f(x)=\sum_{k=0}^{N} c_k \, y_k(x) \qquad (3.3)$$

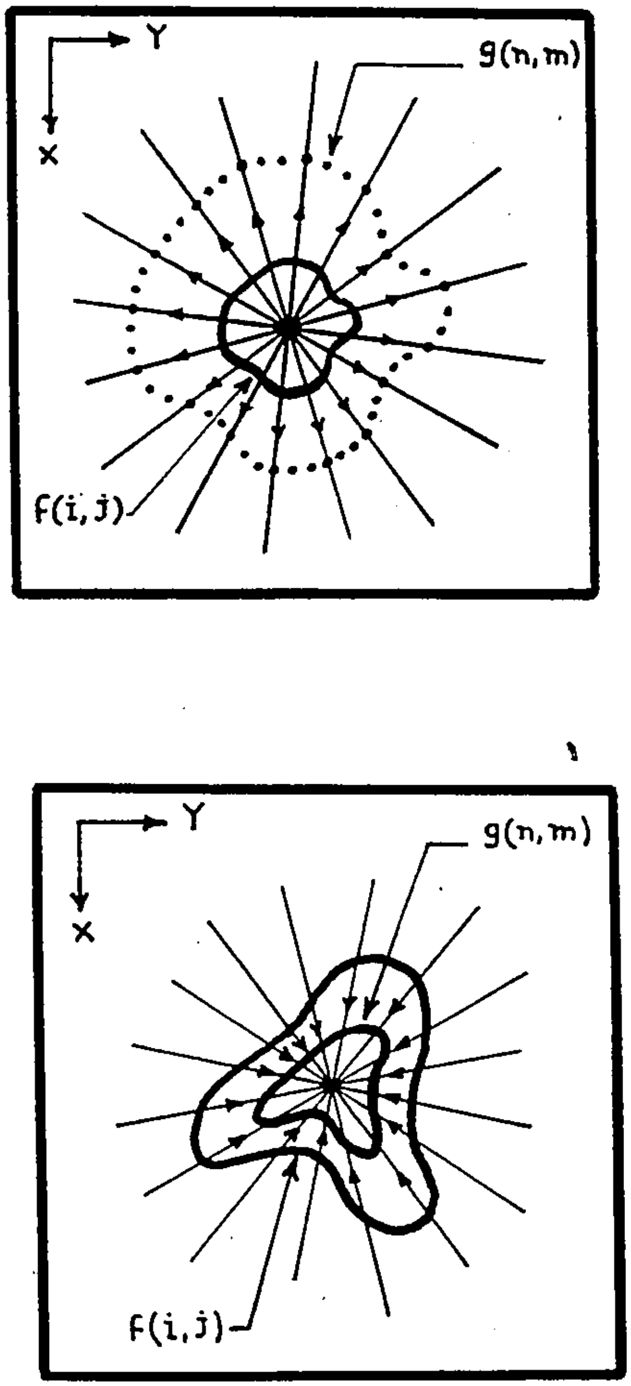


Figure 3.7: ILLUSTRATION OF MINIFICATION AND MAGNIFICATION TRANSFORMATION.

Where c are the coefficients to be determined from the input data, y (x) are the chosen basis functions, and N is the number of given data points.

3.5.2 BASIC_CONCEPT_OF_SPLINE_INTERPOLATION_:

From a numerical point of view, the classical polynomial interpolation approaches [12], e.g., Lagrange interpolation [16], at an increasing set of data points all involve the use of a polynomial of an increasingly higher degree.That approach has several severe limitations. First, it cannot be guaranteed that a sequence of Lagrange interpolations to a continuous function f(x) will converge uniformly to f(x). In fact for any sequence of sets of interpolation points, there exists a continuous function f(x) such that the sequence of lagrange interpolations to f(x) at these points diverges. Second, while the sequence of interpolations may in fact converge to f(x), approximating f'(x) by the derivatives of its interpolation can be extremely inaccurate. These problems can be intuitively linked to two facts concerning polynomial interpolation. First, polynomials have a notorious ability to "wiggle," that is, pinning polynomial down at a few points for a slowly varying function may not produce, in any sense, a good uniform approximation to the function or its derivatives. Second polynomials are analytic functions. Thus, polynomial interpolation is in no sense a "local" procedure. That is, if the function to be interpolated varies rapidly in some part of the region of interest, the effect of this on the interpolation would be felt everywhere.

On the other hand , from the sampling theorem [17] one may attempt to use the Cardinal spline [18] as the basis functions, i.e., let

$$f_k(x) = \frac{\sin 2\pi\Omega(x - x_k)}{2\pi(x - x_K)} \equiv \text{sinc}\,(x - x_K) \qquad (3.4)$$

where Ω is the one-sided bandwith of f(x). And one then concludes that from the sampling the~~o~~rem

$$f(x) = \sum_{k=-\infty}^{\infty} f_k\, y_k(x) \qquad (3.5)$$

is a perfect reconstruction of f(x) if it was originally sampled at or above the Nyquist rate. However, there are many difficulties in doing this because the Cardinal spline, though being analytic, behaves like an infinite degree polynomial whose supports are not local which poses computational problem. If truncations are made on the upper and lower limits of the summation in (3.5), oscillation known as Gibb's phenomenon will show up in $\hat{f}(x)$. Second the interpolation formula in (3.5) has implicity imposed a restriction that the discrete data $\{f_K\}$ must be equally spaced.

The considerations above lead us rather naturally to the idea of interpolating a function by piecewise polynomials, i.e., by analytic functions which are piecewise polynomials of fixed degree. The whole class of piecewise polynomials are called splines. The spline interpolation not only alleviates the difficulties, as it was mentioned previously, suffered by the classical polynomial approach, but also minimizes the least squares errors of the desired function values and its derivatives at the interpolation points. In other words, among the many interpolating functions passing through the data points only the spline interpolation gives the smoothest, which is also the best (in a least square sense) approximation.

3.5.3 Properties of spline basis functions :

In this section we are interested in the B-spline functions [19-26] because they are smooth and span a finite set of data points, i.e, their support is local. Thus they can be used as basis functions in the interpolation formula (3.3). These basis functions can be defined mathematically as follows :

Assume $\pi : x_0 < x_1 \ldots < x_n < x_{n+1}$ is a partition of the interval $[x_0, x_{n+1}]$ on a real axis. A B-spline of degree n on π is, by definition, the following piecewise polynomial :

$$B_n(x;x_0,x_1,x_2,\ldots x_{n+1})=(n+1)\sum_{k=0}^{n+1}\frac{(x-x_k)^n U(x-x_k)}{\omega(x_k)} \tag{3.6}$$

where

$$\omega(x_k)=\prod_{\substack{j=0\\ j\neq k}}^{n+1}(x_k-x_j)$$

$$U(x-x_k)=\begin{cases}(x-x_k)^0 & \text{for } x > x_k\\ 0 & \text{for } x \leqslant x_k\end{cases}$$

$n = 0,1,2,\ldots\ldots$

A sketch of the first four lower order B-splines for the uniformly spaced data points is shown in Fig.(2.3). Evidently the B0 is a sample and hold function such that the interpolation becomes replication. The interpolation by B1 becomes piecewise straight line connections between the knots.Likewise the interpolation by B2 is a graph composed of a sequence of parabolas which join at the knots continuously together with their slopes. Finally the interpolation by B3 is composed of a sequence of third degree piecewise polynomials which join at knots continuously together with their slopes.

It is obvious that the interpolation by B0 and B1 does not yield satisfactory results. On the other hand, when the order of splines increases beyond three, it behaves like normal polynomial interpolation and there is no meaninq of local basis. Therefore, from a smooth interpolation and easy implementation point of view, the cubic spline is a good choice for a basis function.

3.5.4 Properties of cubic spline interpolation :

The cubic spline interpolation [10] was used in this work to interpolate the mapped contour given by its label and its number of points (see Fig.(3.4c)),the result is shown in Fig.(3.4d).

Cubic spline interpolation has the following properties :

1- The curve of the spline between any 2 consecutive data points (weights) is a cubic (y is cubic function).

2-The equation of the slope of the spline between any 2 consecutive data points is a parabola (y' is a parabolic function).

3-The equation of 2nd derivative in the spline between any two consecutive data points is a linear function(y" is linear function).

4-The slope of spline is continuous.Thus, y' values are obtained from the slope equations for any two consecutive spline intervals (defined by 3 data points).

5- y" is the same at the data point common to the 2 consecutive intervals.

We have made the preceding properties looking just at the spline interval between the data points A and B. However the same properties could be made by considering any interval between data points along the spline such as shown in Fig.(3.8).

In a cubic spline fit, it is assumed that the approximating function between any two adjacent data points is a cubic spline regardless of the magnitude of the curvature between the points. Let's us next consider a series of data points (x,y) with i=1,2,3....,n,n+1, where n is the number of data intervales,and determine the equation of the cubic for i-th inteval (the inteval between x and x).Letting i=1,2,3,....,n ,we will obtain a set of cubic equations which will constitute the mathematical model of a spline connecting the data points.

We begin with equation of y" for the i-th interval.Knowing that y" varies linearly over an interval (see Fig.(3.9)), we can write that :

$$y'' = y''_i + \frac{(x - x_i)(y''_{i+1} - y''_i)}{x_{i+1} - x_i} \qquad (3.7)$$

where :

y" :is the second derivative.

$h_i = x_{i+1} - x_i$

The cubic spline function can be given by :

$$y = y''_i \left[\frac{(x_{i+1} - x_i)^3 - h_i(x_{i+1} - x))}{h_i} \right] + \frac{y''_{i+1}}{6} \left[\frac{(x - x_i)^3 - h_i(x - x_i)}{h_i} \right]$$
$$+ y_i \left[\frac{x_{i+1} - x_i}{h_i} \right] + y_{i+1} \frac{[x - x_i]}{h_i} \qquad (3.8)$$

The derivation of this function is given by [4].

$y''_2, y''_3, y''_4, \ldots, y''_n$ are unknown, and can be found by solving the set of equations:

$$\begin{bmatrix} (2h_1 + h_2) & & 0 \cdots\cdots & & & \\ h_2 & 2(h_2 + h_3) & h_3 \cdots\cdots & 0 & 0 & \\ 0 & h_3 & 2(h_3 + h_4) \cdots & 0 & 0 & \\ \vdots & \vdots & \vdots & & & \\ & & h_{n-2} & 2(h_{n-2} + h_{n-1}) & & h_{n-1} \\ \vdots & \vdots & \vdots & & & \\ 0 & 0 & 0 & h_{n-1} & & 2(h_{n-1} + h_n) \end{bmatrix} \begin{bmatrix} y''_2 \\ y''_3 \\ \vdots \\ y''_{n-1} \\ y''_n \end{bmatrix}$$

$$= 6 * \begin{bmatrix} \frac{y_3 - y_2}{h_2} - \frac{y_2 - y_1}{h_1} \\ \frac{y_4 - y_3}{h_3} - \frac{y_3 - y_2}{h_2} \\ \\ \frac{y_{n+1} - y_n}{h_n} - \frac{y_n - y_{n-1}}{h_{n-1}} \end{bmatrix} \tag{3.9}$$

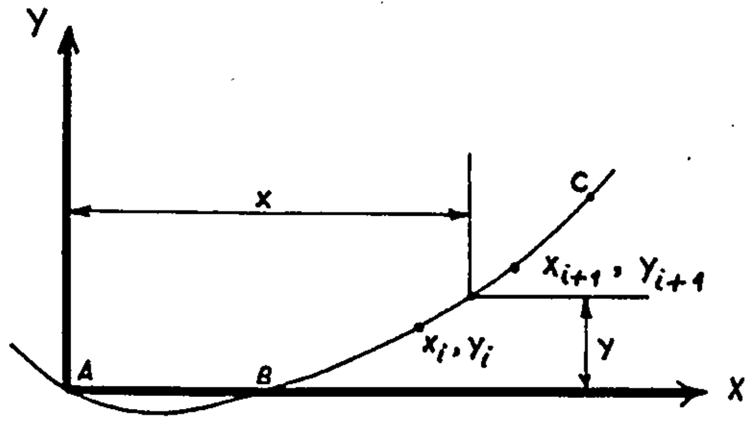


Figure 3.8: A SPLINE PASSING THROUGH 5 DATA POINTS.

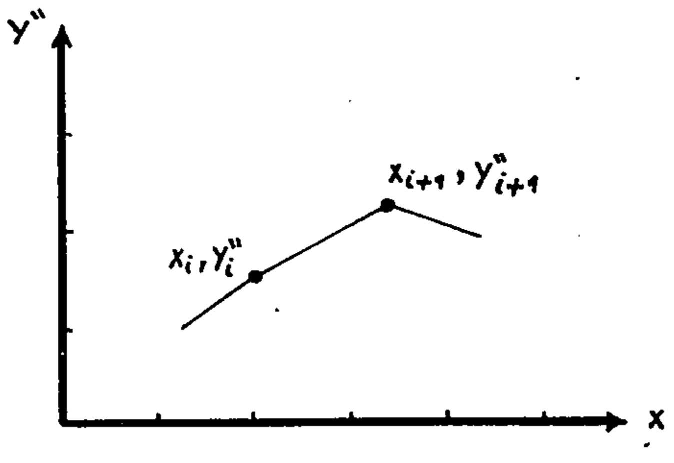


Figure 3.9: A y" CURVE FOR THE ith INTERVAL.

3.5.5 CONTOUR INTERPOLATION :

The contour interpolation is performed in two sub-interpolations : interpolation over the i-axis and interpolation over the j-axis. Before going through the interpolation, a representation of the contour over i and j versus the number of points must be accomplished (Fig.(3.10b) and (3.10c)).Then each representation is interpolated in one dimension separately resulting in Fig.(3.10d) and (3.10e), in the case of one contour. Finally the continuous curves are joined together to form the interpolated contour. This is done by corresponding the first point from the curve inFig.(3.10d) to the first point from the curve in Fig.(3.10e) to form a pixel coordinate. Then the second to second .... etc. The number of representation increases with the number of contours. For example if m is the number of contours, then the number of representation is 2*m.

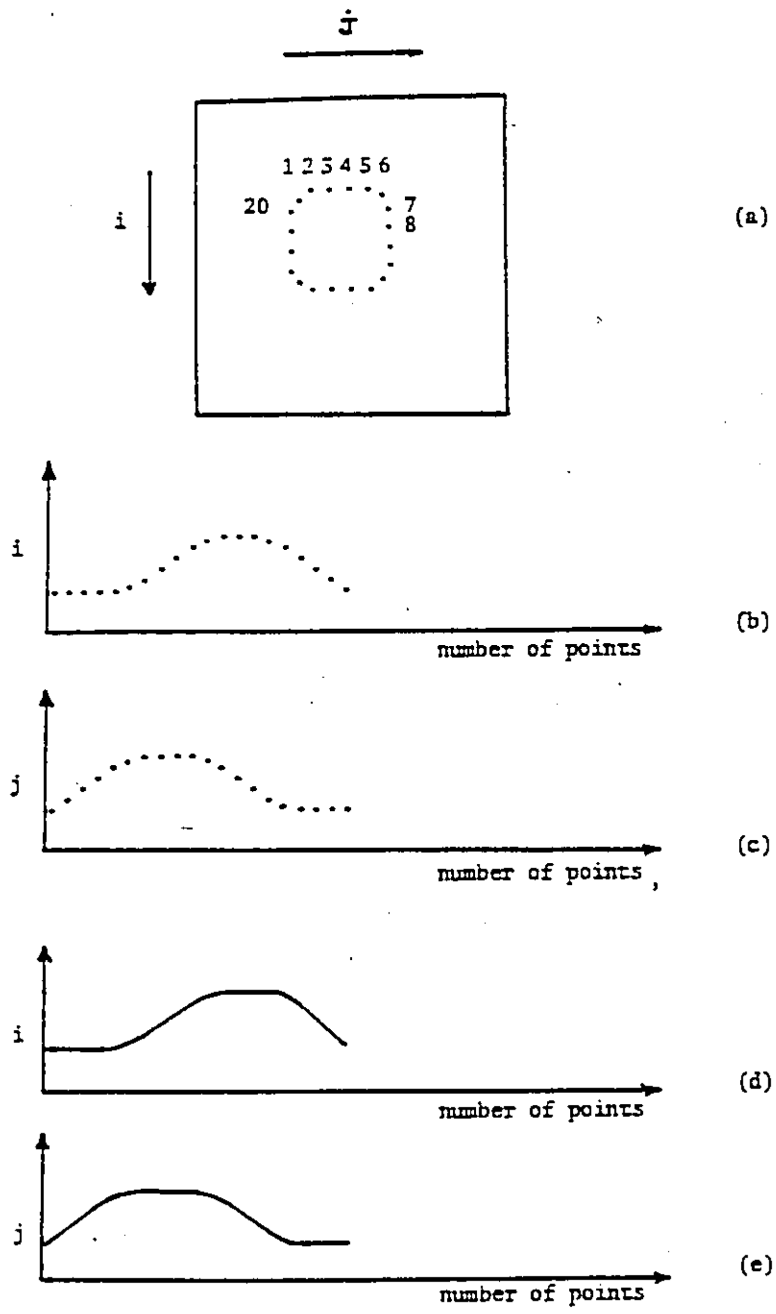


Figure 3.10: ILLUSTRATION OF THE CONTOUR INTERPOLATION.

3.6 LINKING PROCESS :

The only problem faced in this work is the appearance of gaps of one pixel in length (in case of magnification), due to the truncation of the coordinate points, when corresponding an abscissa to its ordinate. The truncation is due to transformation of the pixel coordinates from real in the i and j curves (see Fig.(3.10d) and (3.10e)) to integer in the i-j plane (Fig.(3.10a)). To eliminate this problem, gaps should be filled or linked, for this purpose a linking process is developed.

The linking process used in this work is based on the border following algorithm explained above. and the use of a given template (see Fig.(3.11)). Knowing that the pattern is formed from contours,which can be one or more, the linking is performed as follows:

1-Scan the image row-wise,the first black pixel "1" is labelled different than "0" or "1".

2-Follow the border and label each pixel with value different than "0" or"1".

3-When the next pixel detected in the border is different from "1" ,the preceding pixel (i1,j1) is the first extremity.

4-Use the template to find the other extremity (see Fig.(3.12),by testing the neighbourhood $p_1,p_2,\ldots,p_{16}$ to "1" value.

5-Once the 2nd extremity is detected, one of $p_1, p_2, \ldots, p_8$ is labelled upon the nearest neighbor to the 2nd extremity. Note that the label value must be "1", so it could be detected by the border following.

6-Proceed in the same manner until the first border element detected in step 1 is encountered again.

The procedure explained above is summarized by the flowchart in Fig.(3.13).

3.7 FILLING

3.7.1 INTRODUCTION

In many cases, such as interactive image processing, scene analysis and computer graphics, the problem of extracting or shading a region delineated by a digital contour has to be faced.

A recent paper by PAVLIDIS [27], described a number of algorithms to perform this task. Generally most of these algorithms do not correctly work for non-simple contours.

Fill algorithms are used to change the color of pixels that lie within specified regions. Various algorithms have been developed for displaying filled areas. One method uses the boundary definition to identify which pixels belong to the interior of an area. Boundaries are, in general, lines or curves that define the outer extents of regions. Other methods start from a position within the area and fill outward from this point.

p1 (i-2, j-2)	p2 (i-2, j-1)	p3 (i-2, j)	p4 (i-2, j+1)	p5 (i-2, j+2)
p16 (i-1, j-2)	p1' (i-1, j-1)	p2' (i-1, j)	p3' (i-1, j+1)	p6 (i-1, j+2)
p15 (i, j-2)	p8' (i, j-1)	(i, j)	p4' (i, j+1)	p7 (i, j+2)
p14 (i+1, j-2)	p7' (i+1, j-1)	p6' (i+1, j)	p5' (i+1, j+1)	p8 (i+1, j+2)
p13 (i+2, j-2)	p12 (i+2, j-1)	p11 (i+2, j)	p10 (i+2, j+1)	p9 (i+2, j+2)

Figure 3.11: 5X5 TEMPLATE USED IN THE LINKING.

TABLE

The nearest p_i' for :	p_i'
p_1 and p_2	p_1'
p_3'	p_2'
p_4, p_5 and p_6	p_3'
p_7	p_4'
p_8, p_9 and p_{10}	p_5'
p_{11}	p_6'
p_{12}, p_{13} and p_{14}	p_7'
p_{15}	p_8'
p_{16}	p_1'

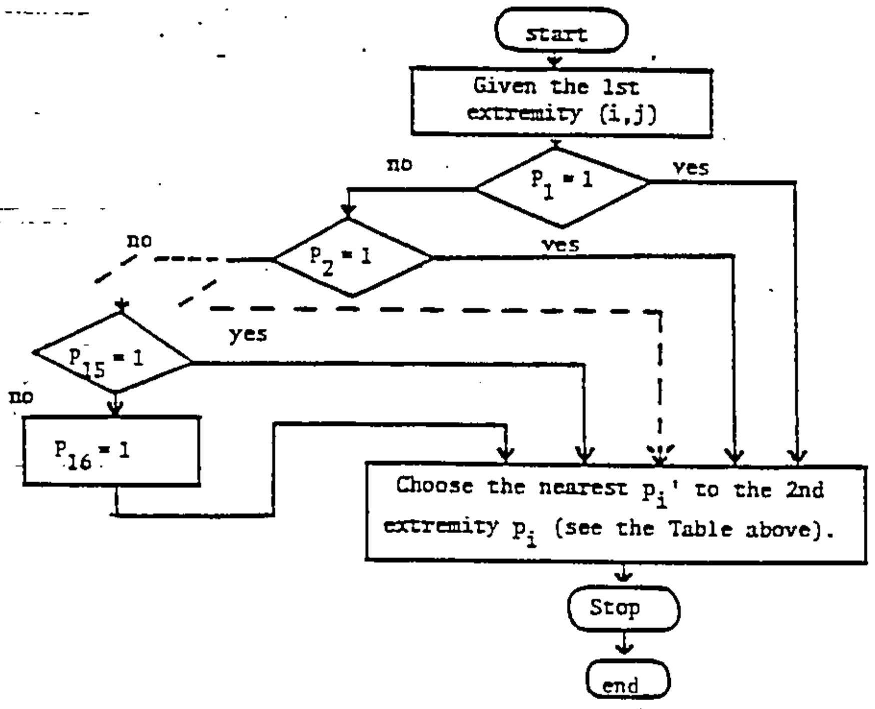

start
Given the 1st extremity (i,j)
$p_1 = 1$
no
yes
$p_2 = 1$
no
yes
$p_{15} = 1$
yes
no
$p_{16} = 1$
Choose the nearest p_i' to the 2nd extremity p_i (see the Table above).
Stop
end

Figure 3-12: NEAREST NEIGHBOR FLOWCHART.

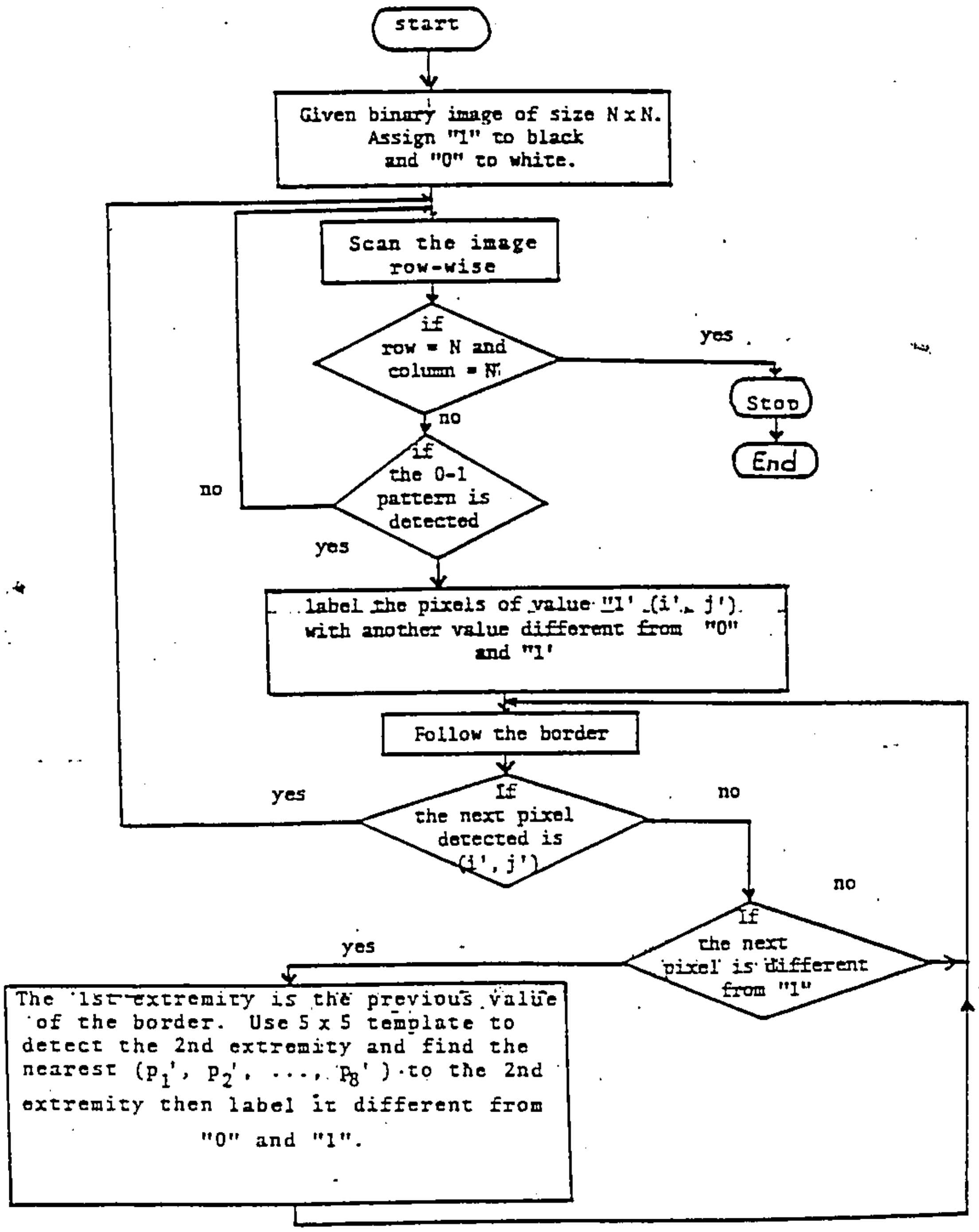

start
Given binary image of size N x N. Assign "1" to black and "0" to white.
Scan the image row-wise
if row = N and column = N
yes
Stop
End
no
if the 0-1 pattern is detected
no
yes
label the pixels of value "1" (i', j') with another value different from "0" and "1"
Follow the border
If the next pixel detected is (i', j')
yes
no
If the next pixel is different from "1"
no
yes
The 1st extremity is the previous value of the border. Use 5 x 5 template to detect the 2nd extremity and find the nearest (p1', p2', ..., P8') to the 2nd extremity then label it different from "0" and "1".

Figure 3.13: FLOWCHART DESCRIBING THE LINKING.

3.7.2 FILLING ALGORITHM [11]

This algorithm performs correctly the filling of every kind of closed digital curve, regardless of its thickness and of the presence of repeating or brush-past arc. Suppose that the contour is represented by "1's" embedded in an NxN array of "0's" and that such an array is scanned row by row, and let x be the row coordinate and y the column coordinate of an element of the array. The algorithm is presented by the following steps :

First step: C is traced by a border following algorithm (B.F.) of the type described in [9] (see appendix A). During B.F. the vectors X and Y are built, where the coordinates of the pixels of the contour are stored in the order they are found by the algorithm.

Second step: The vector X is scanned and for every connected sequence of equal numbers, only the first element of every sequence is retained, together with the corresponding element of the vector Y.

Third step: The vector X is scanned again and for every p_k, starting from p_2, it is checked if it is a contiguous elements, i.e., p_{k-1} and p_{k+1} are such that:

(a) $x_{k-1} > x_k$ and $x_{k+1} < x_k$

or vice versa

(b) $x_{k-1} < x_k$ and $x_{k+1} > x_k$

If (a) or (b) are verified, p_K is saved, otherwise it is erased.

The first element p_1 of vectors must be considered as subsequent to the last one. The total number of saved pixels, that will be called p*'s from now on, must be even.

Fourth step: The p*'s are rearranged within the vectors, according to the increasing value of x and for every x, according to the increasing value of y. Pairs from successive elements starting from one end of the vectors, represent now the extremes of horizontal runs which must be filled by "1's".

3.8 RESULTS AND COMPARISON

The method explained above is summarized by the flowchart shown in Fig.(3.14). The computer simulation results obtained by using this method are shown in Figs.(3.17)-(3.20). As it can be noticed from those figures there is no appearance of jaggies and the edges appear smooth.

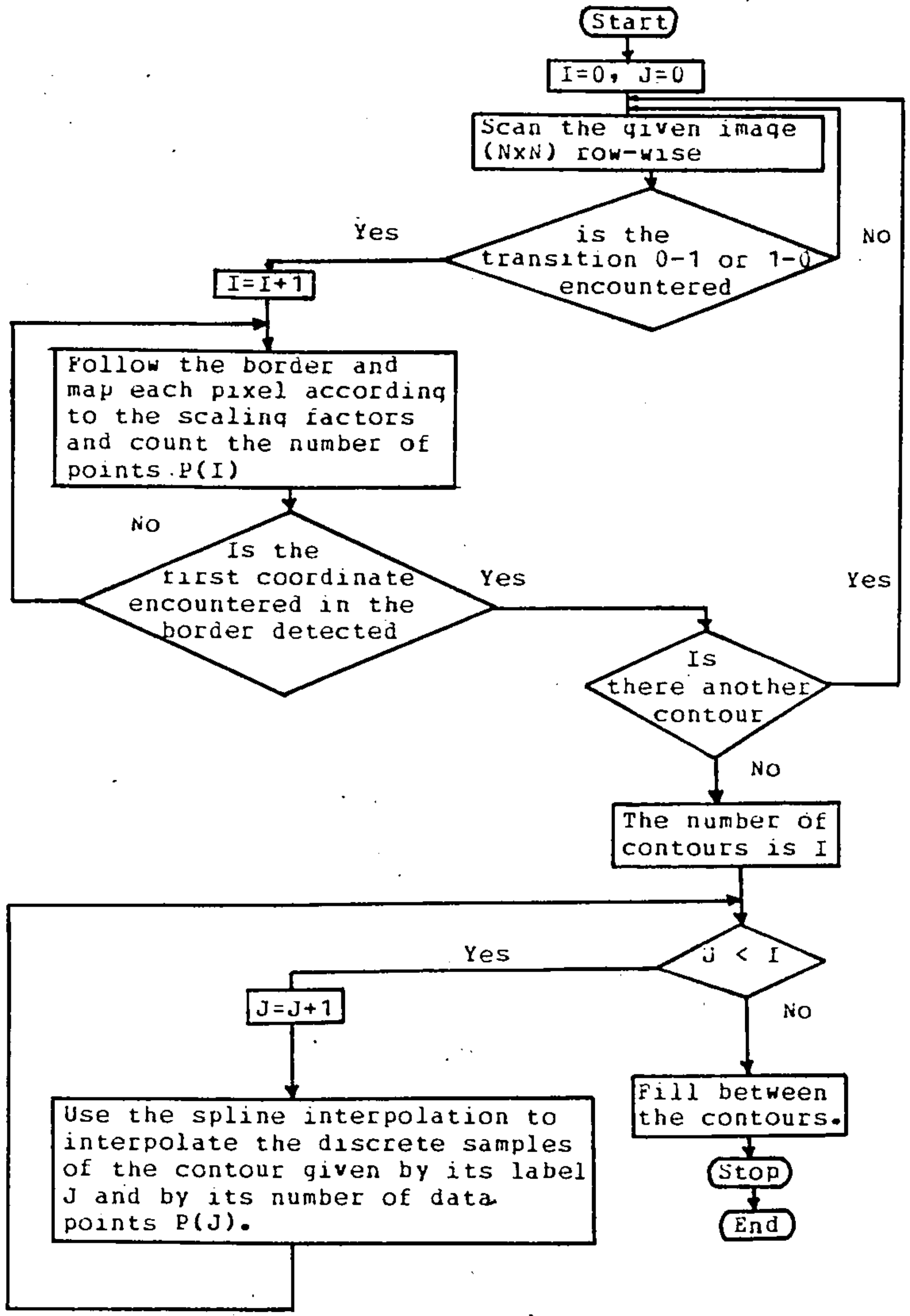

Start
I=0, J=0
Scan the given image (NxN) row-wise
is the transition 0-1 or 1-0 encountered
Yes
No
I=I+1
Follow the border and map each pixel according to the scaling factors and count the number of points P(I)
No
Is the first coordinate encountered in the border detected
Yes
Yes
Is there another contour
No
The number of contours is I
J < I
Yes
No
J=J+1
Use the spline interpolation to interpolate the discrete samples of the contour given by its label J and by its number of data points P(J).
Fill between the contours.
Stop
End

Figure 3.14: FLOWCHART DESCRIBING THE CONTOUR METHOD.

3.9 CHARACTER DESIGN

3.9.1 HISTORIC

The idea of designing letters mathematically goes back to the fifteenth century and it became rather highly developed in the early part of the sixteenth. The design was on capital letters using simple tools such as; ruler and compass. The first person to do this was Felice Feliciano. The Italian mathematician Luca Pacioli has done a lot of work in the design of capital letters. The design of character 'B' in Fig.(3.15) was a part of his work. Apparently nobody carried this work further to lower case letters, numerals, or italic letters and other symbols, until more than 100 years later when Joseph Moxon made a detailed study of some beautiful letters designed in Holland.The generation of typefaces by mathematical means became popular in the seventeeth century, and it was abandoned during the eighteenth century. The twentieth century is the right time to have another look at the generation of typefaces, now that mathematics has advanced and computers are able to do the calculations.

Modern printing equipment based on raster lines in which a metal "type" has been replaced by purely combinatorial patterns of zeros and ones that specify the desired position of ink in a discrete way make mathematics and computer science increasingly relevant to printing.

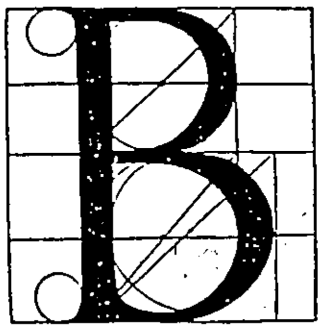


Figure 3.15: Sixteen century ruler-and-compass constructions for the letter B by Pacioli.

Figure 3.16: Generated Arabic character (left) with its key points (right).

They are able to give a completly precise definition of letter shapes that will produce essentially equivalent results on all raster-based machines. Furthermore it is possible to define infinitely many styles of type at once.

3.9.2 Generation of typeface

To explain how to draw a shape, a precise way is needed to specify various key point of that shape. A standard Cartesian coordinate is used for this purpose. The location of a point is defined by specifying its x coordinate, which is the number of units to the right of some reference point, and its y coordinate, which is the number of units upwards from the reference point. In a typical application a rough sketch of the shape is prepared on a piece of a graph paper, and the key points are labelled on that sketch with any convenient numbers. Then a program is written that explains; (i) how to figure out the coordinates of those key points, and (ii) how to draw the desired lines and curves between those points, in this case cubic spline curves are used.

Points are specified in terms of fixed numbers like 300, this means a distance of 300 on the square grid or "raster".

The character shown in Fig.(3.16) was generated on the VAX VT 240 using thirteen key points in the grid of 300x300 units. Two spline curves are used, the first passes through the points A, B, and C, the second through D, E, and F. All the remaining points are joined by straight lines.

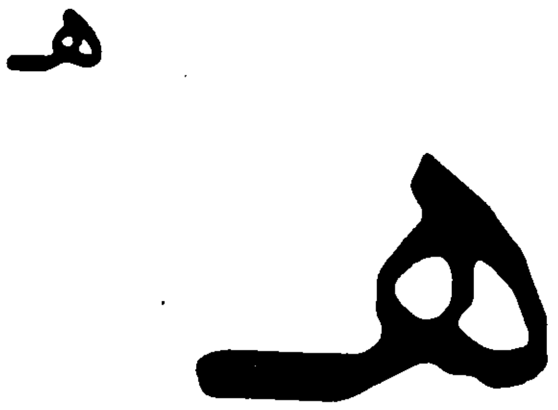


Figure_3.17: Magnification,Contour method for Arabic character,scalx=4.0,scaly=4.0 , character three contours.

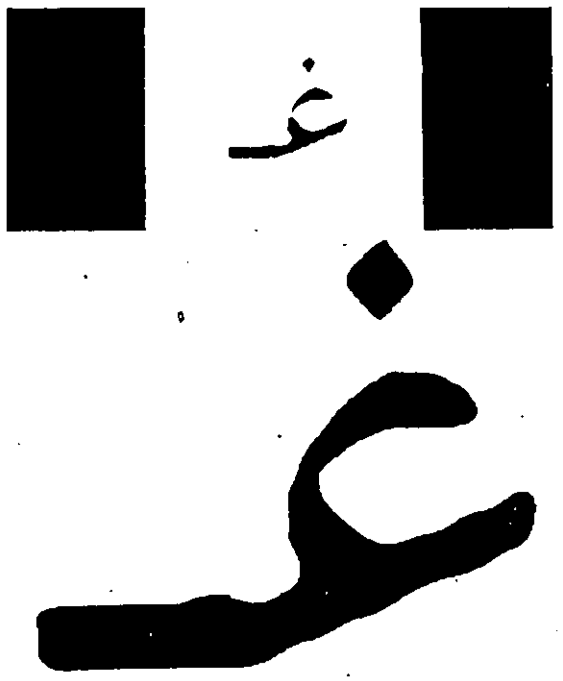


Figure_3.18: Magnification,Contour method for Arabic character,scalx=4.0,scaly=4.0 , Character with dot.

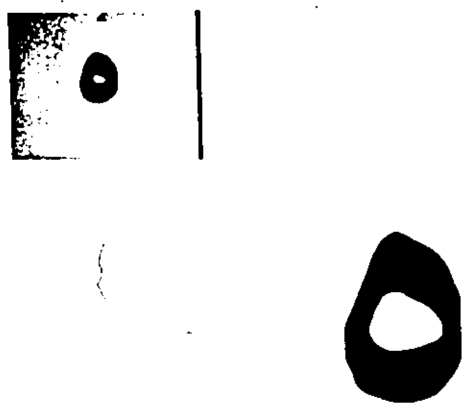


Figure_3.19: Magnification,Contour method for Arabic character,scalx=4.0,scaly=4.0 , Character with two contours.

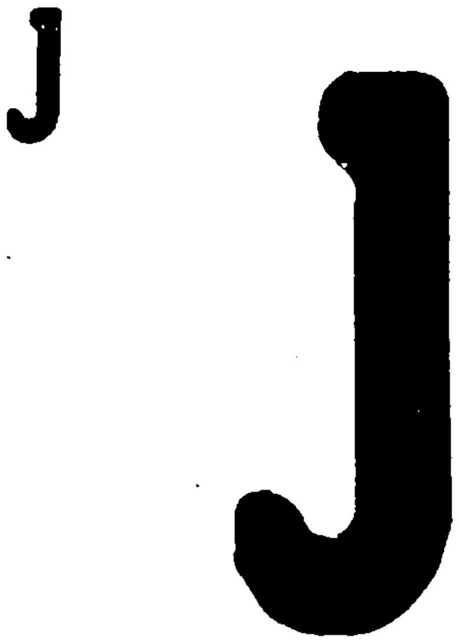


Figure_3.20: Magnification,Contour method for English character,scalx=4.0,scaly=4.0 , Character with one contour.

Chapter IV

COMPARISON AND CONCLUSION

4.1 CHARACTER MAGNIFICATION, RESULTS AND COMPARISON

Three methods have been simulated for character enlargement; these are replication, telescoping template, and the contour method. The computer simulation results obtained by using the above three methods are shown in Figs.(4.1)-(4.3). Among those images in Figs.(4.1b), (4.2b) and (4.3b) are obtained from the replication, in Figs.(4.1c),(4.2c) and (4.3c) from the telescoping template using the third order window (which is the highest order), and in Figs.(4.1d),(4.2d) and (4.3d) from the contour method.

In comparing those results from the different scaling procedure, the replication has resulted in "jaggies" along the edges. In the telescoping template, the character still has jaggies as the quality of the scaled font is dependent on the size of the template. The larger the size, the better the quality and the time requirement increases with the window order as can be shown in Table (2). Time comparison between the replication, the telescoping template and the contour method is given in Table (3). Understandably the superior performance of the contour method is due to the cubic spline fitting that makes the edge of the character smoother.

TABLE 2.

ORDER WINDOW	PROCESSING TIME (image 512x512)
1	2 min. 56 sec.
2	4 min. 10 sec.
3	5 min. 10 sec.

TABLE 3.

	PROCESSING TIME (image 512x512)
Replication method	3 min. 90 sec.
Telescoping template method (3rd order)	5 min. 10 sec.
Contour method	3 min. 30 sec.

a b
c d

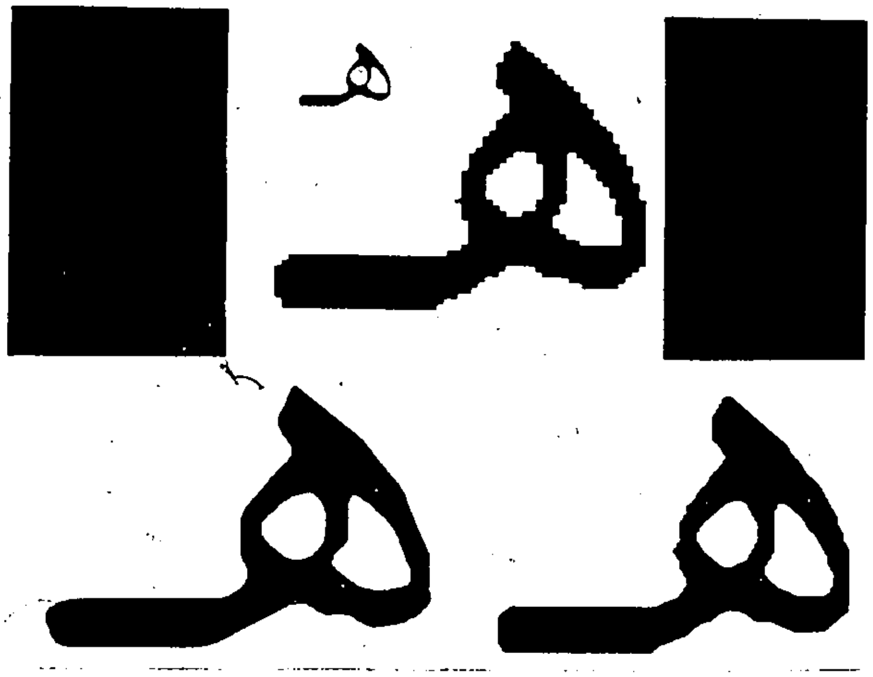


Figure 4.1: (a) Given binary image,magnification; scalx=4.0,scaly=4.0 (b) Replication, (c) Telescoping template method, (d) Contour method.

a b
d c

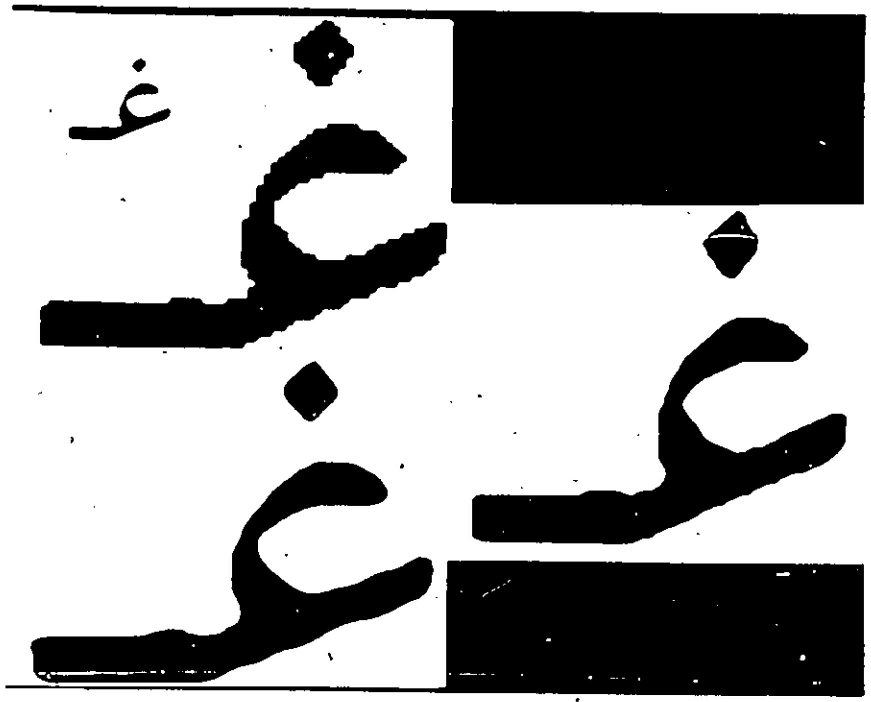


Figure 4.2: (a) Given binary image,magnification; scalx=4.0,scaly=4.0 (b) Replication, (c) Telescoping template method, (d) Contour method.

b c
a
d

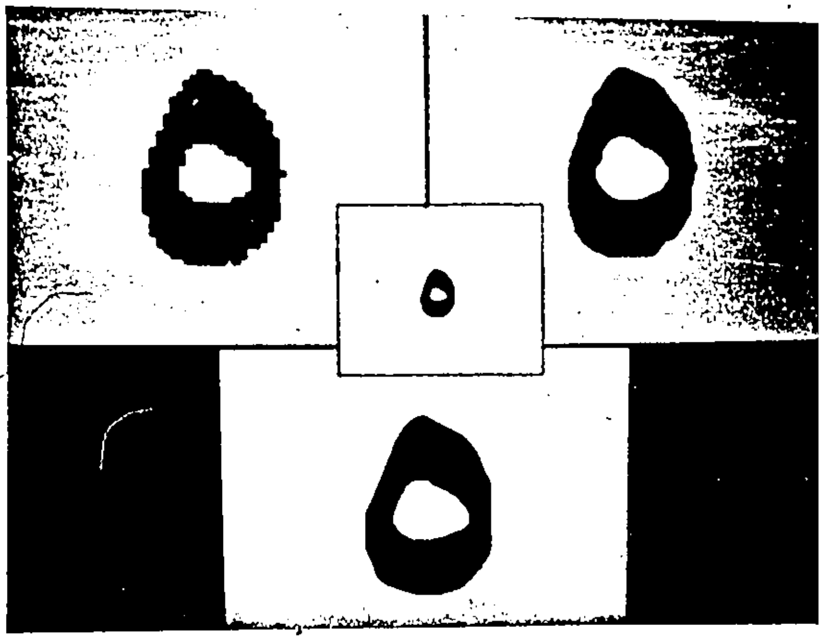


Figure_4.3: (a) Given binary image,magnifica scalx=4.0,scaly=4.0 (b) Replication, Telescoping template method, (d) Co method.

4.2 Character minification: results

Some results for character minification using this method are shown in Figs.(4.4)-(4.9).

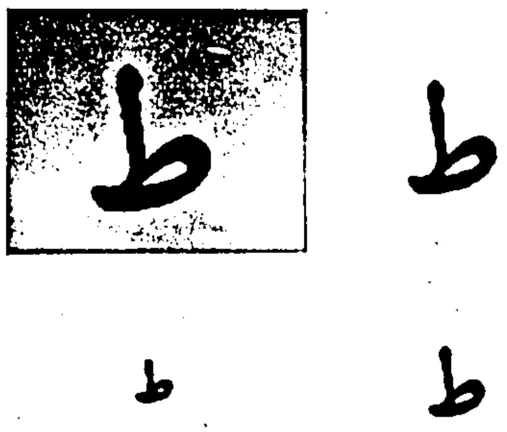


Figure_4.4: Minification; a) Given Arabic character, b) scalx=0.8, scaly=0.8 c)scalx=0.5, scaly=0.5 d) scalx=0.3, scaly=0.3 .

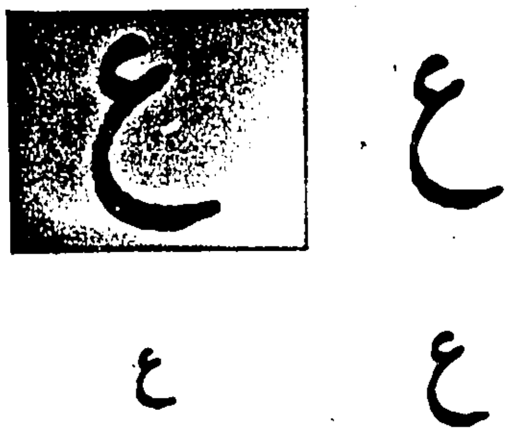


Figure_4.5: Minification; a) Given Arabic character, b) scalx=0.8, scaly=0.8 c)scalx=0.5, scaly=0.5 d) scalx=0.3, scaly=0.3 .

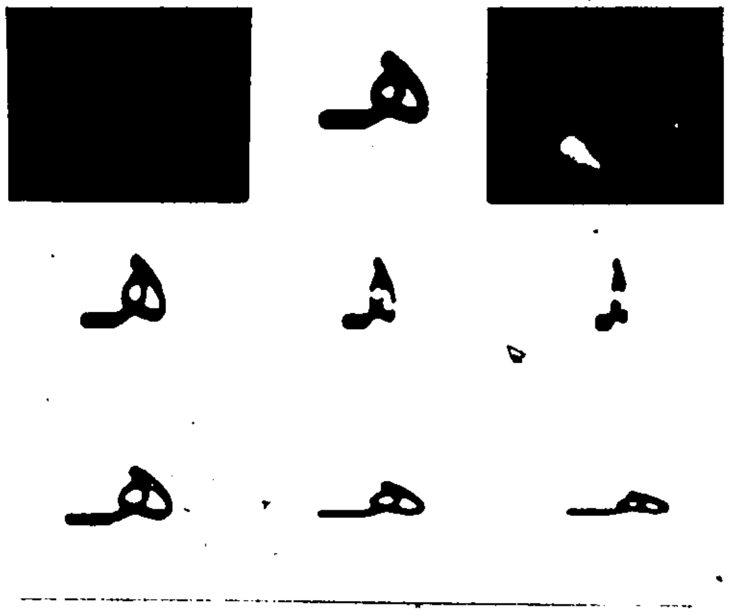


Figure 4.6: Minification; a) Given Arabic character, b) scalx=1.0, scaly=0.8 c) scalx=1.0, scaly=0.5 d) scalx=1.0, scaly=0.3 e) scalx=0.8, scaly=1.0 f) scalx=0.5, scaly=1.0 g) scalx=0.3, scaly=1.0 .

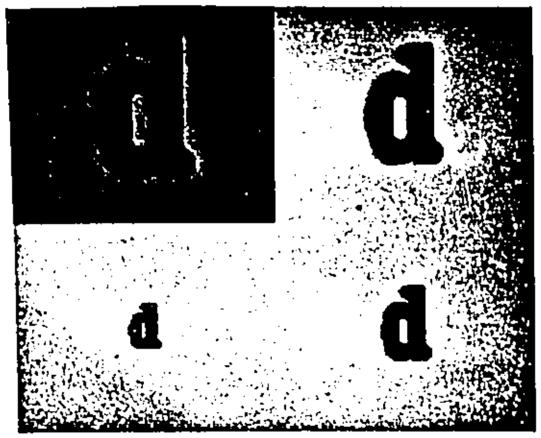


Figure 4.7: Minification; a) Given English character, b) scalx=0.8, scaly=0.8 c) scalx=0.5, scaly=0.5 d) scalx=0.3, scaly=0.3 .

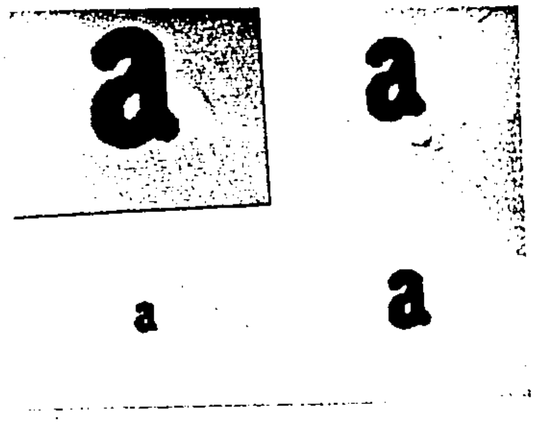


Figure_4.8: Minification; a) Given English character, b) scalx=0.7, scaly=0.7 c)scalx=0.5, scaly=0.5 d) scalx=0.25, scaly=0.25 .

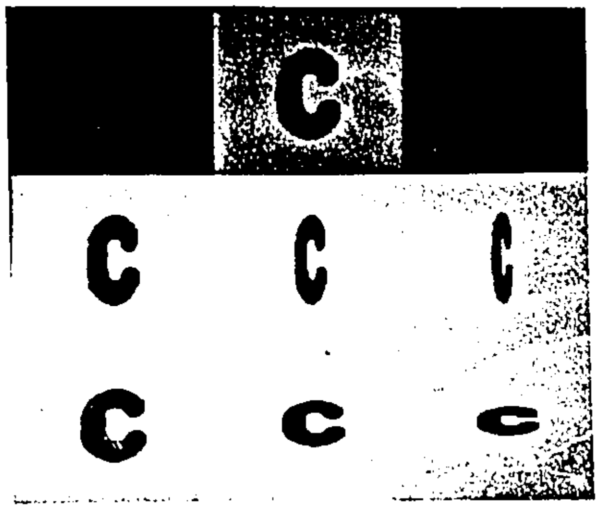


Figure_4.9: Minification; a) Given English character, b) scalx=1.0, scaly=0.8 c)scalx=1.0, scaly=0.5 d) scalx=1.0, scaly=0.3 e) scalx=0.8, scaly=1.0 f) scalx=0.5, scaly=1.0 g) scalx=0.3, scaly=1.0 .

4.3 CONCLUSION :

The contour method described above was performed according to the block diagram given in Fig.(3.1). The result was reasonnably excellent, and the jaggies produced by the methods shown above ,were eliminated. This method allows work to be done on the contours only instead of working on the whole image. The main advantage of this method is that the binary image can be covered by its borders alone.

REFERENCES

1. R. A. Ulichney and D. E. Troxel "Scaling of binary images with telescoping template" IEEE Trans. PAMI.4, No.3, pp.331-335, 1982.

2. R. A. Ulicheny "Digital scaling of binary images", M.S. Thesis Massachusetts Inst. Technology, Cambridge, Jan.1979.

3. W. Pratt "Digital image processing", Vol.1, 1978.

4. H. S. Hou "Cubic spline for image interpolation and digital filtering", IEEE Trans. Acoust., Speech, Signal Processing, Vol. ASSP-26, No.6, pp.508-517, Dec. 1978.

5. R. G. Keys, "Cubic convolution for digital image processing", IEEE Trans. Acoust., Speech, Signal processing, Vol.ASSP-29, No. 6, pp.1153-1160, Dec. 1981.

6. A. D. Kulkarni and K. Sivaraman, "Interpolation of digital imagery using hypersurface approximation", Signal processing 7, North-holland, pp.65-73, 1984.

7. Donald E. Knuth, TEX and METAFONT, New Directions in Typesetting. Digital Press, 1979.

8. J. S. Wezska, R. N. Nagel, and A. Rosenfeld "A threshold selection technique" IEEE Trans. on Comput., Vol.C-23, pp.1322-1326, 1974.

9. A. Chottera, and M. Shridhar "Feature extraction of manufactured parts in the presence of spurious surface reflections", Can. Elect. Eng. J. Vol 7 No.4, pp.29-33, 1982.

10. M. L. James, GM. Smith and J. C. Wolford, "Applied numerical methods for digital computation", Harper and Row, 3rd edition, 1985.

11. L. P. Cordella and Di Paolo, "Contour filling for region extraction", North-Holland Publishing Company, Signal Processing 3, pp.247-252, 1981.

12. R. W. Shafer and L. Rabiner, "_A_digital_signal processing_to_interpolation_", Proc. IEEE, Vol.61, pp.692-702, June 1973.

13. R. C. Gonzalez, and P. Wintz "_Digital_image processing_", Addison-Wesley publishing company, Inc., 1977.

14. R. Rajendran "_Surface_flow_detection_using_digital image_processing_techniques_", Master Thesis, Univ. of Windsor, Dept. of Elect. Eng., 1984.

15. M. A. Janqua "_Image_thresholding_and_feature extraction_techniques_", Master Thesis, Univ of Windsor, Dept. Enq., 1982.

16. E. K. Blum "_Numerical_analysis_and_computation_: Theory_and_practice_", Reading MA : Addison-wesley, 1972.

17. C. E. Shannon "_Communication_in_the_presence_of_noise ", Proc. IRE, Vol. 37, pp.10-21, Jan. 1949.

18. E. T. Whittaker "_On_the_functions_which_are represented_by_the_expansions_of_the_interpolation theory_", Proc. Roy. Soc. Edinburgh, Vol. 35, pp.181-194, 1915.

19. T. N. E. Greville "_Theory_and_application_of_spline functions_", New-York : Academic, 1969.

20. M. Schultz "_Spline_Analysis_", Englewood Cliffs, NJ: Prentice-hall, 1973.

21. T. N. E. Greville "_Spline_functions_and_its_minimal properties_", In Proceeding of the Conference on Approximation Theory, P. L. Butzer and J. Korevaar, Eds. Basel:Verlag, 1974.

22. T. N. E. Greville "_Spline_functions, Interpolation and_numerical_quadrature", In Mathematical Methods for Digital Computer, Vol.2, A. Ralston and H. S. Wilf, Eds. New York: Wiley, 1967.

23. T. Lynche and L. Schumaker "_Computation_of_smoothing and_interpolating_natural_splines_via_local_basis_", Center of numerical analysis, University of Texas, Austin , Rep. 17 ,Ap. 1974.

24. C. Deboor "_On_uniform_approximation_by_splines_", J. Approximation Theory, Vol.1, pp.219-235, 1968.

25. J. H. Ahlberg, E. N. Nilson, and J. L. Walsh "_The theory_of_splines_and_their_applications_", New York : Academic, 1967.

26. M. J. Munteau, and L. L. Schumaker "_Some multi-dimentional_spline_approximation_methods_", J. Approximation Theory, Vol.10, pp.23-40, 1974.

27. T. Pavlidis, ,Filling_algorithms_for_raster_graphics, Vol. 10, 1979, pp. 126-141.

APPENDIX A

Border following algorithm :

The border following algorithm used here is an adaptation of the method described in reference 5.These steps should be followed:

1- Detect the first border element at (i1,j1) (a dark pixel) through a row (or column) scan.The element immediately preceding (i1,j1) is labelled as the first neighbor (id,jd).

2- Starting with (id,jd) and proceeding clockwise,label the other seven neighbors of (i1,j1) as 2,3,....,8. set k=2.

3- Evaluate the coordinates lx(k),ly(k) of the k neighbor of (i1,j1) using the table shown below.

4- If the pixel at the k is a '1' (i.e.,a dark pixel),then this pixel is the next border element.Define (i1,j1) as this element and (id,jd) as the preceding element.Go to step 2.

5-If the pixel at the k neighbor is a '0',set k=k+1 and goto step 3.

6- Proceed until the first border element detected in step 1 is encountered again.

TABLE 1

Co-ordinates of eight neighbours

	LX(J)	LY(J)
1	ID	JD
2	LX(1)+K1	LY(1)-K2
3	LX(2)-K2	LY(2)-K1
4	LX(3)-K2	LY(3)-K1
5	LX(4)-K1	LY(4)+K2
6	LX(5)-K1	LY(5)+K2
7	LX(6)+K2	LY(6)+K1
8	LX(7)+K2	LY(7)+K1

Co-ordinates of border element(i1,j1)
Co-ordinates of first neighbour(id,jd)

K1 = JD -J1 K2 = ID -I1

IF |K |=1, |K |=1, K=0

```
c***************  APPENDIX (B)
*****************************c
c- This program performs the scaling of binary images using
c  contour method. It goes through different steps as follows;
c  1) Border detection using border following. 2) Scaling
c  mapping.3) Contour interpolation over i and j axis.
c  4) Linking process .5) filling in between the contours.
c
************************************************************
c
c
        implicit real*8(a-h,o-z)
        dimension y(160),f(160),a(160),b(160),c(160),d(160)
        real scalx,scaly
        integer h(512,512),g(512,512),ai(600,4),aj(600,4),
     * pcount(4)

        integerlx(8),ly(8),count,label,nsize,max1(2)
        integer y1(1300),y2(1300),x1(260),x2(260),scal,max
        character img(512,512)
        character*1 gg(128)
        character*16 filn,film
        write(*,*)'enter the input filename ------->'
        read(*,'(a16)') filn
        write(*,*)'enter the input size image :'
        read(*,*) nsize
        write(*,*)'enter the scaling factor over x and y '
        read(*,*) scalx,scaly
        write(*,*)'enter the scaling factor '
        read(*,*) scalx
       open(1,file=filn,recl=128,form='formatted',status=
     * 'old')
        do 1 i=1,nsize
        do 1 j=1,nsize/128
        read(1,456) gg
 456    format(128a)
        do 457 k=1,128
        h(i,(j-1)*128+k)=ichar(gg(k))
 457    continue
 1      continue
        close(1)
        do 10 i=1,nsize
        do 10 j=1,nsize
        if(h(i,j).eq.0) then
        h(i,j)=1
        else
        h(i,j)=0
        endif
 10     continue

c--------- Border detection , to extract each contour of
           the character

        do 67 i=1,nsize
        do 67 j=1,nsize
        g(i,j)=255
 67     continue
        count=0
        label=0
        do 20 i=1,nsize-1
        do 20 j=1,nsize-1
        if(h(i,j-1).eq.0.and.h(i,j).eq.1) goto 100
        goto 20
 100    count=count+1
        k=0
        i1=i
        j1=j
        id=i
        jd=j-1
        ibeg=i1
        jbeg=j1
 25     call cord(lx,ly,i1,j1,id,jd)
c*********************************
        do 30 jj=2,8
        n=lx(jj)
        m=ly(jj)
        if(h(n,m).eq.1) then
        i1=lx(jj)
        j1=ly(jj)
        id=lx(jj-1)
        jd=ly(jj-1)
        k=k+1
        h(i1,j1)=100
        pcount(count)=k

c------- Scaling mapping is performed on each contour pixel.
```

```
        nn=int(scalx*real(i1)+256.0*(1.0-scalx))
        mm=int(scaly*real(j1)+256.0*(1.0-scaly))
        ai(k,count)=nn
        aj(k,count)=mm
        g(nn,mm)=0
        if(i1.eq.ibeg.and.j1.eq.jbeg) then
        h(i1,j1)=100
        g(nn,mm)=0
        goto 20
        else
        goto 25
        endif
        endif
 30     continue
c***************************
 20     continue
        do 99 i=1,count
        write(*,*)' number of pts = ',pcount(i)
        write(*,*)`in the',i,' contour'
 99     continue
c###############################################
        do 1958 i=1,nsize
        do 1958 j=1,nsize
        h(i,j)=255
 1958   continue
c###############################################
        write(*,*) 'enter kk'
        read(*,*) kk

c------- Contour interpolation over i and j axis.

        do 343 k=1,count
        m=1
        do 11 i=1,pcount(k)
        if(m.le.pcount(k)) goto 999
        goto 995
 999    x1(i)=aj(m,k)
        x2(i)=ai(m,k)
        goto 996
 995    nmax=i-1
        goto 112
 996    m=m+kk
```

```
11      continue
112     n=nmax
        x1(nmax+1)=x1(1)
        x2(nmax+1)=x2(1)
c**************************'
        write(*,*) 'n=','' ',n
c*****************************************
        m=1
        do 19 i=1,nmax+1
        y(i)=real(m)
        m=m+scal*kk
19      continue
        write(*,*) 'y(n)=',' ',y(nmax+1)
        n=nmax+1
c*****************************************
        do 32 ii=1,2
        if(ii.eq.1) then
        do 14 i=1,n
        f(i)=real(x1(i))
14      continue
        else
        do 15 i=1,n
        f(i)=real(x2(i))
15      continue
        endif
        call spln(n,y,f,a,b,c,d)
        lk=1300
        hh=1.0
        x=y(1)-hh
        do 8 i=1,lk
        x=x+hh
        if(x.gt.y(n)) goto 32
        call sple(n,y,a,b,c,d,x,p)
        if(ii.eq.1) then
        y1(i)=int(p+0.5)
        else
        y2(i)=int(p+0.5)
        endif
8       continue
32      continue
        max=int(y(n)),
        write(*,*) 'max=',' ',max
        do 126 i=1,max
```

```
      h(y2(i),y1(i))=0
 126  continue
 343  continue
      do 129 i=1,nsize
      do 129 j=1,nsize
      if(h(i,j).eq.0) then
      h(i,j)=1
      else
      h(i,j)=0
      endif
 129  continue

c---- Linking the gaps (due to real-integer transformation).
        using template 5 x 5.

      count=0
      label=0
      do 2 i=1,nsize-1
      do 2 j=1,nsize-1
      if(h(i,j-1).eq.0.and.h(i,j).eq.1) goto 3
      goto 2
 3    count=count+1
      k=0
      i1=i
      j1=j
      h(i1,j1)=100
      id=i
      jd=j-1
      ibeg=i1
      jbeg=j1
 4    call cord(lx,ly,i1,j1,id,jd)
c*********************************
      do 5 jj=2,8
      n=lx(jj)
      m=ly(jj)
      if(h(n,m).eq.1) then
      i1=lx(jj)
      j1=ly(jj)
      id=lx(jj-1)
      jd=ly(jj-1)
      k=k+1
      h(i1,j1)=100
      pcount(count)=k
```

```
          if(i1.eq.ibeg.and.j1.eq.jbeg) then
          h(i1,j1)=100
          goto 2
          else
          goto 4
          endif
          endif
 5       continue
c******************************
c 4 neighbours
c
          if(h(i1,j1-2).eq.1) then
          h(i1,j1-1)=1
          goto 4
          endif
          if(h(i1,j1+2).eq.1) then
          h(i1,j1+1)=1
          goto 4
          endif
          if(h(i1-2,j1).eq.1) then
          h(i1-1,j1)=1
          goto 4
          endif
          if(h(i1+2,j1).eq.1) then
          h(i1+1,j1)=1
          goto 4
          endif
c************************************
c corner neibours
c
          if(h(i1-2,j1+2).eq.1.or.h(i1-2,j1+1).eq.1.or.
     *h(i1-1,j1+2).eq.1) then
          h(i1-1,j1+1)=1
          goto 4
          endif
          if(h(i1+2,j1+2).eq.1.or.h(i1+1,j1+2).eq.1.or.
     *h(i1+2,j1+1).eq.1) then
          h(i1+1,j1+1)=1
          goto 4
          endif
          if(h(i1+2,j1-2).eq.1.or.h(i1+2,j1-1).eq.1.or.
     *h(i1+1,j1-2).eq.1) then
          h(i1+1,j1-1)=1
```

```
           goto 4
           endif
           if(h(i1-2,j1-2).eq.1.or.h(i1-2,j1-1).eq.1.or.
        *h(i1-1,j1-2).eq.1) then
           h(i1-1,j1-1)=1
           goto 4
           endif
 2         continue
 400       continue
           do 6 i=1,nsize
           do 6 j=1,nsize
           if(h(i,j).eq.100) then
           h(i,j)=0
           else
           h(i,j)=255
           endif
 6        continue
c************************************************************
c************************************************************

c-- ----- Filling in between the contours.

           do 29 i=2,nsize
           k=0
           mm=255
           nn=0
           do 29 j=2,nsize
           if(h(i,j-1).eq.255.and.h(i,j).eq.0)then
           k=k+1
           max1(k)=j
           endif
           if(k.eq.2) then
           do 1911 n=max1(1),max1(2)
           h(i,n)=0
 1911      continue
           k=0
           endif
 29        continue
           do 1964 i=2,nsize-1
           do 1964 j=2,nsize-1
           if(h(i,j).eq.0.and.h(i-1,j).eq.255.and.h(i+1,j).eq.
         * 255) then
           h(i,j)=255
         endif
         if(h(i,j).eq.255.and.h(j-1,j).eq.0.and.h(i+1,j).eq.
        *0) then
         h(i,j)=0
         endif
 1964    continue
c*******************************************************
        open(2,file='out.img',recl=128,recordtype='fixed',
      * form='formatted',status='new',carriagecontrol='none')
        do 81 i=1,nsize
        do 81 j=1,nsize/128
        do 108 k=1,128
        nx=int(h(i,(j-1)*128+k))
        gg(k)=char(nx)
 108    continue
        write(2,456) gg
 456     format(128a)
 81      continue
         close(2)
         stop
         end


c*********************************************
         subroutine spln(n,x,f,a,b,c,d)
c*********************************************
         implicit real*8(a-h,o-z)
         dimension x(n),f(n),a(n),b(n),c(n),d(n),h(250),
       *t(250,251),u(250)
         nm1=n-1
         nm2=n-2
         do 1 i=1,nm1
         h(i)=x(i+1)-x(i)
         u(i)=(f(i+1)-f(i))/h(i)
         a(i)=f(i)
 1       continue
         do 2 i=1,nm2
         do 2 j=1,nm2
         t(i,j)=0.d0
 2       continue
         do 3 i=1,nm2
         t(i,i)=2.d0*(h(i)+h(i+1))
```

```
3       continue
        if(n.gt.3) then
        do 4 i=2,nm2
        t(i,i-1)=h(i)
        t(i-1,i)=h(i)
4       continue
        endif
        do 5 i=1,nm2
        t(i,nm1)=3.d0*(u(i+1)-u(i))
5       continue
        n2=n-2
        m=1
        nd=250
        ndpm=nd+m
        eps=0.0000001d0
        call gaus1(n2,m,nd,ndpm,t,eps)
        do 6 i=2,nm1
        c(i)=t(i-1,nm1)
6       continue
        c(1)=0.d0
        c(n)=0.d0
        do 7 i=1,nm1
        b(i)=u(i)-h(i)*(2.d0*c(i)+c(i+1))/3.d0
        d(i)=(c(i+1)-c(i))/(h(i)*3.d0)
7       continue
        return
        end

c*********************************************
        subroutine sple(n,x,a,b,c,d,t,p)
c*********************************************
        implicit real*8(a-h,o-z)
        dimension x(n),a(n),b(n),c(n),d(n)
        i=2
6       if(t.gt.x(i)) go to 5
        i=i-1
        go to 7
5       i=i+1
        go to 6
7       continue
        t1=t-x(i)
        p=a(i)+t1*(b(i)+t1*(c(i)+d(i)*t1))
        return
```

```
      end
c*****************************************
      subroutine gaus1(n,m,nd,ndpm,a,delt)
c*****************************************
      implicit real*8(a-h,o-z)
      dimension a(nd,ndpm)
      nm1=n-1
      if(n.gt.1) then
      do 1 k=1,nm1
      u=dabs(a(k,k))
      kk=k+1
      in=k
      do 2 i=kk,n
      if(dabs(a(i,k)).gt.u) then
      u=dabs(a(k,k))
      in=i
      endif
2     continue
      mpn=m+n
      if(k.ne.in) then
      do 3 j=k,mpn
      x=a(k,j)
      a(k,j)=a(in,j)
      a(in,j)=x
3     continue
      endif
      if(u.lt.delt) then
      write(6,4)
4     format(2x,'the matrix is singular.Gaussian,
     * elimination cannot be performed.')
      return
      endif
      do 5 i=kk,n
      do 5 j=kk,mpn
      a(i,j)=a(i,j)-a(i,k)*a(k,j)/a(k,k)
5     continue
1     continue
      if(dabs(a(n,n)).lt.delt) then
      write(6,4)
      return
      endif
      do 6 k=1,m
      a(n,k+n)=a(n,k+n)/a(n,n)
      do 6 ie=1,nm1
      i=n-ie
      ix=i+1
      do 7 j=ix,n
      a(i,k+n)=a(i,k+n)-a(j,k+n)*a(i,j)
 7    continue
      a(i,k+n)=a(i,k+n)/a(i,i)
 6    continue
      return
      else if(dabs(a(1,1)).lt.delt) then
      write(6,4)
      return
      endif
      do 8 j=1,m
      a(1,n+j)=a(1,n+j)/a(1,1)
 8    continue
      return
      end
c*******************************************
      subroutine cord(lx,ly,i1,j1,id,jd)
c*******************************************
      dimension lx(8),ly(8)
      k1=jd-j1
      k2=id-i1
      lx(1)=id
      lx(2)=lx(1)+k1
      lx(3)=lx(2)-k2
      lx(4)=lx(3)-k2
      lx(5)=lx(4)-k1
      lx(6)=lx(5)-k1
      lx(7)=lx(6)+k2
      lx(8)=lx(7)+K2
      ly(1)=jd
      ly(2)=ly(1)-k2
      ly(3)=ly(2)-k1
      ly(4)=ly(3)-k1
      ly(5)=ly(4)+k2
      ly(6)=ly(5)+k2
      ly(7)=ly(6)+k1
      ly(8)=ly(7)+k1
      return
      end
```

```
c**************  APPENDIX (C)  **************************
c
c       This program performs the replication method.
c
c*********************************************************
        integer h(128,128),g(128,128),fact
        character img(128,128)
        integer nsize,nn,mm,n,m,scalx,scaly
        character*16 filnme
        write(*,*)' enter the filename '
        read(*,'(a16)') filnme
        write(*,*)' enter the size of the image ---->'
        read(*,*) nsize
        write(*,*)' enter the scaling factors over X and Y '
        read(*,*) scalx,scaly
        write(*,*)' 1'
        open(1,file=filnme,form='binary',status='old')
        open(2,file='out.img',form='binary',status='new')
        write(*,*)' 2'
        do 10 i=1,nsize
        do 10 j=1,nsize
        g(i,j)=0
 10     continue
        do 20 i=1,nsize
        do 20 j=1,nsize
        read(1) img(i,j)
        h(i,j)=ichar(img(i,j))
        if(h(i,j).eq.0)then
        h(i,j)=1
        else
        h(i,j)=0
        endif
 20     continue
        do 30 i=1,nsize
        do 30 j=1,nsize
        kk=h(i,j)+h(i,j+1)+h(i+1,j+1)+h(i+1,j)
        n=scalx*i+64*(1-scalx)
        m=scaly*j+64*(1-scaly)
        if(n.gt.128.or.n.lt.0.or.m.gt.128.or.m.lt.0) goto 30
        nn=scalx+n-1
        mm=scaly+m-1
        if(kk.eq.0.or.kk.eq.1.or.kk.eq.2) goto 30
        if(kk.eq.4) goto 100
```

```
      goto 30
100   continue
      do 40 k=n,nn
      do 40 l=m,mm
      g(k,l)=1
40    continue
30    continue
      do 90 i=1,nsize
      do 90 j=1,nsize
      if(g(i,j).eq.1) then
      g(i,j)=0
      else
      g(i,j)=255
      endif
90    continue
      do 200 i=1,nsize
      do 200 j=1,nsize
      img(i,j)=char(g(i,j))
200   continue
      write(2) ((img(i,j),j=1,nsize),i=1,nsize)
      close(2)
      stop
      end
```

```
c****************  APPENDIX (D)  ************************
c
c     This program performs the telescoping template method
c     with the first order window .
c
c*********************************************************
        integer h(512,512),g(512,512),fact
        integer nsize,size,nn,mm,n,m,scalx,scaly,range
        character*1 gg(128)
        character*16 filn,film
        write(*,*)'enter the input filename'
        read(*,'(a16)') filn
        write(*,*)'enter the output filename'
        read(*,'(a16)') film
        write(*,*)'enter the input size image :'
        read(*,*) nsize
        write(*,*)'enter the scaling factors over x and y :'
        read(*,*) scalx,scaly
        open(1,file=filn,recl=128,form='formatted',status=
     * 'old')
        do 761 i=1,nsize
        do 761 j=1,nsize/128
        read(1,456) gg
 456    format(128a)
        do 457 k=1,128
        h(i,(j-1)*128+k)=ichar(gg(k))
        457continue
 761    continue
        close(1)
c*************************
       do 10 i=1,nsize
       do 10 j=1,nsize
       g(i,j)=0
 10    continue
       do 20 i=1,nsize
       do 20 j=1,nsize
       if(h(i,j).eq.0)then
       h(i,j)=1
       else
       h(i,j)=0
       endif
20    continue
       do 30 i=1,nsize
```

```
      do 30 j=1,nsize
      kk=h(i,j)+h(i,j+1)+h(i+1,j+1)+h(i+1,j)
      n=scalx*i+256*(1-scalx)
      m=scaly*j+256*(1-scaly)
      nn=scalx+n-1
      mm=scaly+m-1
      if(kk.eq.0.or.kk.eq.1.or.kk.eq.2) goto 30
      range=n+m+scalx-1
      if(kk.eq.4) goto 200
      goto 300
200   continue
      do 40 k=n,nn
      do 40 l=m,mm
      g(k,l)=1
40    continue
300   continue
      if(kk.eq.3) goto 400
      goto 30
400   continue
      if(h(i,j).eq.0) goto 500
      goto 600
500   continue
      do 50 k=n,nn
      do 50 l=m,mm
      if(k+l.ge.range) then
      g(k,l)=1
      endif
50    continue
600   continue
      if(h(i+1,j+1).eq.0) goto 700
      goto 800
700   continue
      do 60 k=n,nn
      do 60 l=m,mm
      if(k+l.le.range) then
      g(k,l)=1
      endif
60    continue
800   continue
      if(h(i+1,j).eq.0) goto 900
      goto 5
900   continue
      do 70 k=n,nn
```

```
        do 70 l=m,mm
        if(n.ge.m) goto 1
        dif=m-n
        if(k+dif.le.l) then
        g(k,l)=1
        endif
        goto 70
1       continue
        dif=n-m
        if(l+dif.ge.k) then
        g(k,l)=1
        endif
70      continue
5       continue
        if(h(i,j+1).eq.0) goto 1100
        goto 30
1100    continue
        do 80 k=n,nn
        do 80 l=m,mm
        if(n.ge.m) goto 2
        dif=m-n
        if(k+dif.ge.l) then
        g(k,l)=1
        endif
        goto 80
2       continue
        dif=n-m
        if(l+dif.le.k) then
        g(k,l)=1
        endif
80      continue
30      continue
        do 90 i=1,nsize
        do 90 j=1,nsize
        if(g(i,j).eq.1) then
        h(i,j)=0
        else
        h(i,j)=255
        endif
90      continue
        open(2,file=film,recl=128,recordtype='fixed',
     * form='formatted',status='new',carriagecontrol='none')
        do 81 i=1,nsize
```

```
      do 81 j=1,nsize/128
      do 108 k=1,128
      nx=int(h(i,(j-1)*128+k))
      gg(k)=char(nx)
108   continue
      write(2,456) gg
81    continue
      close(2)
      stop
      end
```

```
c****************  APPENDIX (E) ****************************c
c      This program performs the telescoping template method
c      with the second and third order window.
c
c***********************************************************
        integer h(512,512),g(512,512),fact
        integer nsize,size,nn,mm,n,m,scalx,scaly,range
        character*1 gg(128)
        character*16 filn,film
        write(*,*)'enter the input filename'
        read(*,'(a16)') filn
        write(*,*)'enter the output filename'
        read(*,'(a16)') film
        write(*,*)'enter the input size image :'
        read(*,*) nsize
        open(1,file=filn,recl=128,form='formatted',status='old'
        do 761 i=1,nsize
        do 761 j=1,nsize/128
        read(1,456) gg
 456    format(128a)
        do 457 k=1,128
        h(i,(j-1)*128+k)=ichar(gg(k))
 457    continue
 761    continue
        close(1)
c*************************
        do 20 i=1,nsize
        do 20 j=1,nsize
        if(h(i,j).eq.0)then
        h(i,j)=1
        g(i,j)=1
        else
        h(i,j)=0
        g(i,j)=0
        endif
 20     continue
        do 30 i=3,nsize-2
        do 30 j=3,nsize-2
        if(h(i,j).eq.1) then
        p1=h(i-2,j-1)
        p2=h(i-2,j)
        p3=h(i-2,j+1)
```

```
      p4=h(i-1,j-1)
      p5=h(i-1,j)
      p6=h(i-1,j+1)
      p7=h(i,j-1)
      p8=h(i,j)
      p9=h(i,j+1)
      p10=h(i+1,j-1)
      p11=h(i+1,j)
      p12=h(i+1,j+1)
      p13=h(i+2,j-1)
      p14=h(i+2,j)
      p15=h(i+2,j+1)
      p16=h(i-1,j+2)
      p17=h(i,j+2)
      p18=h(i+1,j+2)
      p19=h(i-1,j-2)
      p20=h(i,j-2)
      p21=h(i+1,j-2)
c
      s1=p1+p2+p4+p5+p7
      s2=p3+p6+p9+p12+p15+p14+p11+p10+p13
      if(s1.eq.0.and.s2.eq.9) then
      g(i-2,j)=1
      g(i-1,j)=1
      g(i,j-1)=1
      goto 30
      endif
      s1=p2+p3+p5+p6+p9
      s2=p1+p4+p7+p10+p13+p11+p14+p12+p15
      if(s1.eq.0.and.s2.eq.9) then
      g(i-2,j)=1
      g(i-1,j)=1
      g(i,j+1)=1
      goto 30
      endif
      s1=p9+p12+p15+p11+p14
      s2=p1+p2+p3+p4+p5+p6+p7+p10+p13
      if(s1.eq.0.and.s2.eq.9) then
      g(i+2,j)=1
      g(i+1,j)=1
      g(i,j+1)=1
      goto 30
      endif
      s1=p7+p10+p13+p11+p14
      s2=p1+p2+p3+p4+p5+p6+p9+p12+p15
      if(s1.eq.0.and.s2.eq.9) then
      g(i+2,j)=1
      g(i+1,j)=1
      g(i,j-1)=1
      goto 30
      endif
c*****************************************************
      s1=p20+p7+p11+p10+p21
      s2=p19+p4+p5+p6+p16+p9+p17+p18+p12
      if(s1.eq.0.and.s2.eq.9) then
      g(i,j-2)=1
      g(i,j-1)=1
      g(i+1,j)=1
      goto 30
      endif
      s1=p20+p7+p5+p4+p19
      s2=p21+p10+p11+p6+p16+p9+p17+p18+p12
      if(s1.eq.0.and.s2.eq.9) then
      g(i,j-2)=1
      g(i,j-1)=1
      g(i-1,j)=1
      goto 30
      endif
c
      s1=p5+p6+p16+p9+p17
      s2=p19+p4+p20+p7+p21+p10+p11+p12+p18
      if(s1.eq.0.and.s2.eq.9) then
      g(i,j+2)=1
      g(i,j+1)=1
      g(i-1,j)=1
      goto 30
      endif
c
      s1=p11+p12+p18+p9+p17
      s2=p19+p4+p20+p7+p21+p10+p5+p6+p16
      if(s1.eq.0.and.s2.eq.9) then
      g(i,j+2)=1
      g(i,j+1)=1
      g(i+1,j)=1
      goto 30
      endif
```

```
c
      endif
30    continue
      do 90 i=1,nsize
      do 90 j=1,nsize
      if(g(i,j).eq.1) then
      h(i,j)=0
      else
      h(i,j)=255
      endif
90    continue
      open(2,file=film,recl=128,recordtype='fixed',
     * form='formatted',status='new',carriagecontrol='none')
      do 81 i=1,nsize
      do 81 j=1,nsize/128
      do 108 k=1,128
      nx=int(h(i,(j-1)*128+k))
      gg(k)=char(nx)
108   continue
      write(2,456) gg
81    continue
      close(2)
      stop
      end
```

```
c***************  APPENDIX (F)  *************************
c
c        This program performs the generation of an
c        Arabic character, using 13 key points.
c         The character was generated on the VAX/ VT 240.
c
c*******************************************************
        implicit real*8(a-h,o-z)
        dimension x(10),y(10),xx(10),yy(10),p0(10),p1(10),
     * p2(10),p3(10)
        open(4,file='sou.dat',status='new')
        write(4,*) char(27)//'P0p'
        call init
        call erasecolor(0)
        n=3
c*******************************
        data(x(i),i=1,3)/100.,160.,300./
        data(y(i),i=1,3)/100.,250.,200./
        call coeff(x,y,n,p0,p1,p2,p3)
        x1=99.0
        do 11 i=1,201
        x1=x1+1.0
        call slope(n,x,p0,p1,p2,p3,x1,p)
        if(x1.lt.150.) goto 11
        if(x1.eq.150.) then
        print*,p
        endif
        if(x1.eq.180.) then
        print*,'first'
        print*,x1,p
        endif
        call move(int(x1),int(p))
        call drawl(int(x1),int(p))
 11     continue
c*******************************
        data(xx(i),i=1,3)/100.,160.,250./
        data(yy(i),i=1,3)/100.,230.,200./
        call coeff(xx,yy,n,p0,p1,p2,p3)
        x1=99.0
        do 12 i=1,151
        x1=x1+1.0
        call slope(n,xx,p0,p1,p2,p3,x1,p)
        if(x1.lt.150.) goto 12
        if(x1.eq.150.) then
        print*,p
        endif
        if(x1.eq.180.) then
        print*,x1,p
        endif
        call move(int(x1),int(p))
        call drawl(int(x1),int(p))
 12     continue
        call coline(150.,151.,233.333,214.814814)
        call coline(250.,100.,200.,200.)
        call coline(100.,101.,200.,180.)
        call coline(100.,300.,180.,180.)
        call coline(300.,301.,180.,200.)
c Dot :
        call coline(200.,215.,150.,165.)
        call coline(215.,230.,165.,150.)
        call coline(230.,215.,150.,135.)
        call coline(215.,200.,135.,150.)
        call move(int(200.+300.),int(150.))
        call drawl(int(200.+300.),int(150.))
        call move(int(215.+300.),int(165.))
        call drawl(int(215.+300.),int(165.))
        call move(int(230.+300.),int(150.))
        call drawl(int(230.+300.),int(150.))
        call move(int(215.+300.),int(135.))
        call drawl(int(215.+300.),int(135.))
        call move(int(150.+300.),int(233.))
        call drawl(int(150.+300.),int(233.))
        call move(int(150.+300.),int(214.))
        call drawl(int(150.+300.),int(214.))
        call move(int(250.+300.),int(200.))
        call drawl(int(250.+300.),int(200.))
        call coline(150.,151.,233.333,214.814814,a,b)
        call coline(250.,100.,200.,200.,a,b)
        call coline(100.,101.,200.,180.,a,b)
        call coline(100.,300.,180.,180.,a,b)
        call coline(300.,301.,180.,200.,a,b)
        call move(int(100.+300.),int(200.))
        call drawl(int(100.+300.),int(200.))
        call move(int(100.+300.),int(180.))
        call drawl(int(100.+300.),int(180.))
        call move(int(300.+300.),int(180.))
        call drawl(int(300.+300.),int(180.))
        call move(int(300.+300.),int(200.))
        call drawl(int(300.+300.),int(200.))
        call move(int(180.+300.),int(261.))
        call drawl(int(180.+300.),int(261.))
        call move(int(180.+300.),int(231.))
        call drawl(int(180.+300.),int(231.))
        WRITE(4,*) CHAR(27)//'
        CLOSE(4)
        stop
        end
c****************************
        subroutine slope(n,x,p0,p1,p2,p3,t,p)
        implicit real*8(a-h,o-z)
        dimension x(n),p0(n),p1(n),p2(n),p3(n)
        i=2
 6      if(t.gt.x(i)) go to 5
        i=i-1
        go to 7
 5      i=i+1
        go to 6
 7.     continue
        t1=t-x(i)
        p=p0(i)+p1(i)*(t1)+p2(i)*(t1)**2+p3(i)*(t1)**3
        return
        end
c******************************************
        subroutine coeff(x,y,n,p0,p1,p2,p3)
        implicit real*8(a-h,o-z)
        dimension x(20),y(20),m(20),t(20),p0(n),p1(n),
     * p2(n),p3(n)
        do 10 i=1,n-1
        m(i)=(y(i+1)-y(i))/(x(i+1)-x(i))
 10     continue
```

```
      if(m(2).eq.m(1).and.m(n-1).eq.m(n-2)) then
      t(1)=(m(n-1)+m(1))/2
      else
      t(1)=(abs(m(2)-m(1))*m(n-1)+abs(m(n-1)-m(n-2))*m(1))
     * /(abs(m(2)-m(1))+abs(m(n-1)-m(n-2)))
      endif
      if(m(3).eq.m(2).and.m(n-1).eq.m(1)) then
      t(2)=(m(n-1)+m(2))/2
      else
      t(2)=(abs(m(3)-m(2))*m(1)+abs(m(1)-m(n-1))*m(2))/
     * (abs(m(3)-m(2))+abs(m(1)-m(n-1)))
      endif
      if(m(n-3).eq.m(n-2).and.m(n-1).eq.m(1)) then
      t(2)=(m(n-1)+m(n-2))/2
      else
      t(n-1)=(abs(m(1)-m(n-1))*m(n-2)+abs(m(n-2)-m(n-3))*
     * m(n-1))/(abs(m(1)-m(n-1))+abs(m(n-2)-m(n-3)))
      endif
      do 20 i=3,n-2
      if(m(i+1).eq.m(i).and.m(i-1).eq.m(i-2)) then
      t(i)=(m(i)+m(i-2))/2
      else
      t(i)=(abs(m(i+1)-m(i))*m(i-1)+abs(m(i-1)-m(i-2))
     * m(i))/(abs(m(i+1)-m(i))+abs(m(i-1)-m(i-2)))
      endif
20    continue
      do 30 i=1,n-1
      p0(i)=y(i)
      p1(i)=t(i)
      p2(i)=(3*(y(i+1)-y(i))/(x(i+1)-x(i))-2*t(i)-t(i+1))/
     * (x(i+1)-x(i))
      p3(i)=(t(i)+t(i+1)-2*(y(i+1)-y(i))/(x(i+1)-x(i)))/
     *(x(i+1)-x(i))**2
```

```
 30    continue
       return
       end
c*****************************************************
       subroutine coline(x1,x2,y1,y2)
       real x1,x2,y1,y2,a,b
       xx=x1-x2
       yy=y1-y2
       a=yy/xx
       b=(x1*y2-x2*y1)/xx
c************************************************************
       if(abs(xx).ge.abs(yy)) then
       if(x1.gt.x2) then
       k=-1
       else
       k=1
       endif
       do 10 i=int(x1),int(x2),k
       x=real(i)
       call line(x1,y1,x2,y2,a,b,x,y)
       call move(int(x),int(y))
       call drawl(int(x),int(y))
 10    continue
       else
       if(y1.gt.y2) then
       k=-1
       else
       k=1
       endif
       do 20 i=int(y1),int(y2),k
       y=real(i)
       call line(x1,y1,x2,y2,a,b,y,x)
```

```
      call move(int(x),int(y))
      call drawl(int(x),int(y))
 20   continue
      endif
      return
      end
c***************************************
      subroutine line(x1,y1,x2,y2,a,b,x,y)
      real a,b,x,y,x1,y1,x2,y2
      xx=abs(x1-x2)
      yy=abs(y1-y2)
      if(xx.ge.yy) then
      y=a*x+b
      else
      y=x/a-b/a
      endif
      return
      end
```

```
C***********  APPENDIX (G)  *********************
C
C        GRAPHIC SUBROUTINES
C
C**************************************************
C****************************
        SUBROUTINE INIT
C****************************
        WRITE(4,10)
10      FORMAT(' S(A[0,470] [799,0])')
        RETURN
        END
C****************************
        SUBROUTINE ERASECOLOR(P)
C****************************
        INTEGER P
        WRITE(4,10) P
10      FORMAT(' S(I',I1,',','E) ')
        RETURN
        END
C****************************
        SUBROUTINE ERASESC
C****************************
        WRITE(4,10)
10      FORMAT(' S(E) ')
        RETURN
        END
C****************************
        SUBROUTINE MOVE(X,Y)
C****************************
        INTEGER X,Y
        WRITE(4,10) X,Y
10      FORMAT(' P[',I3,',',I3,']')
        RETURN
        END
C****************************
        SUBROUTINE DRAWL(X1,Y1)
C****************************
        INTEGER X1,Y1
        WRITE(4,20) X1,Y1
20      FORMAT(' V[',I3,',',I3,']')
        RETURN
        END
```

```
C**************************
        SUBROUTINE DRAWC(X1,Y1)
C**************************
        INTEGER X1,Y1
        WRITE(4,20) X1,Y1
20      FORMAT(' C[',I3,',',I3,']')
        RETURN
        END
C**************************
        SUBROUTINE DRAWCX(X1)
C**************************
        INTEGER X1
        WRITE(4,20) X1
20      FORMAT(' C[',I3,']')
        RETURN
        END
C**************************
        SUBROUTINE DRAWCY(Y1)
C**************************
        INTEGER Y1
        WRITE(4,20) Y1
20      FORMAT(' C[',',',I3,']')
        RETURN
        END
C**************************
        SUBROUTINE DRAWLX(X1)
C**************************
        INTEGER X1
        WRITE(4,20) X1
20      FORMAT(' V[',I3,']')
        RETURN
        END
C**************************
        SUBROUTINE DRAWLV(X1)
C**************************
        INTEGER X1
        WRITE(4,20) X1
20      FORMAT(' V[','+',I3,']')
        RETURN
        END
C**************************
        SUBROUTINE DRAWLY(Y1)
C**************************
```

```
      INTEGER Y1
      WRITE(4,20) Y1
20    FORMAT(' V[',',',I3,']')
      RETURN
      END
C***************************
      SUBROUTINE SIZETXT(SIZE)
C***************************
      INTEGER SIZE
      WRITE(4,10) SIZE
10    FORMAT(' T(S<',I2,'>)')
      RETURN
      END
C***************************
      SUBROUTINE MESSAGE(MESS)
C***************************
      CHARACTER*15 MESS
      WRITE(4,10) MESS
10    FORMAT(' T" ',A9,' "')
      RETURN
      END
C***************************
      SUBROUTINE MAP
C***************************
      WRITE(4,10)
10    FORMAT(' S(M0(AD)1(AB)2(AR)3(AG)) ')
      RETURN
      END
C***************************
      SUBROUTINE FORCOLOR(P)
C***************************
      INTEGER P
      WRITE(4,10) P
10    FORMAT(' W(I<',I1,'>)')
      RETURN
      END
C***************************
      SUBROUTINE SHADEON
C***************************
      WRITE(4,10)
10    FORMAT(' W(S1)')
      RETURN
      END
```

```
C**************************
        SUBROUTINE SHADEOFF
C**************************
        WRITE(4,*)
10      FORMAT(' W(SO)')
        RETURN
        END
C***************************
        SUBROUTINE SHADEY(M)
C***************************
        INTEGER M
        WRITE(4,30) M
30      FORMAT(' W(S[',',',I3,'])')
        RETURN
        END
C***************************
        SUBROUTINE SHADEX(N)
C***************************
        INTEGER N
        WRITE(4,30) N
30      FORMAT(' W(S(X)[',',',I3,'])')
        RETURN
        END
C****************************
       SUBROUTINE DRAWSEC(X,Y,N)
C***************************
        INTEGER X(20),Y(20),N
        WRITE(4,10) (X(I),Y(I),I=1,N)
10      FORMAT(' C(S)',20('[',I4,',',I4,']'),'(E)')
        RETURN
        END
```

VITA AUCTORUS

June 10th 1960	Born in Dirah (BOUIRA) ALGERIA
June 1980	Completed High School (Baccalaureat) Lycee Abane Ramdane (Algiers) ALGERIA
September 1986	Admitted in the Master's program Department of Electrical Engineering University of Windsor.
June 1988	Candidate for the degree of M.A.Sc. Electrical Engineering University of Windsor Windsor Ontario, CANADA N9B 3P4