Image Expansion using Segmentation-based method

by

Abdul Karim MURAD AGHA

B.Sc. AL-ISRA University, JORDAN, 1996

A THESIS SUBMITTED IN PARTIAL FULFILLMENT OF

THE REQUIREMENTS FOR THE DEGREE OF

MASTER OF APPLIED SCIENCE

in

THE FACULTY OF GRADUATE STUDIES

DEPARTMENT OF ELECTRICAL AND COMPUTER ENGINEERING

We accept this thesis as conforming

to the required standard

THE UNIVERSITY OF BRITISH COLUMBIA

November 2000

In presenting this thesis in partial fulfilment of the requirements for an advanced degree at the University of British Columbia, I agree that the Library shall make it freely available for reference and study. I further agree that permission for extensive copying of this thesis for scholarly purposes may be granted by the head of my department or by his or her representatives. It is understood that copying or publication of this thesis for financial gain shall not be allowed without my written permission.

Department of Electrical & Computer Eng.

The University of British Columbia
Vancouver, Canada

Date Nov 8th, 2000

Abstract

Image enlargement is a vital process for printing images on large format printers. Image expansion introduces many new pixels to the original image. Proper values for these new pixels should be found otherwise distortion and significant degradation in the quality of the image will result. Several methods have been employed to minimize such distortion. The most popular of these methods are the pixel replication and the linear interpolation methods. This is because these methods are not computationally demanding.

In this work, we develop a new image expansion method that preserves the quality of the edges in the expanded image, where other methods fail. First, the edges in the original image are extracted. For binary images we use Laplacian operator and for gray-level images we use Canny edge detector. We propose a new algorithm, which replicates the edge shape so that the edges in the expanded image do not appear zigzagged. Our algorithm expands the edges while it preserves the edge information such as the shape and the continuity of the edge. The edge-shape-replication algorithm simply, classifies the edges into two classes: short post-CCC and long post-CCC edges, each is expanded differently. Most of the original pixels keep their values after the expansion. However, the remaining pixels' values are calculated using one of three proposed techniques based

on the location of the pixel whose value is to be calculated. The three locations of these pixels are pixels that lie on an edge, pixels that are adjacent to edges, and the remaining pixels areas.

To compare performance of our proposed method with others, we reduced the test images to their quarter size then expanded them to their original sizes. The sum of absolute errors (SAD's) between the original image and the expanded images are calculated. Our proposed method is shown to outperform the replication, linear and cubic interpolation methods.

Contents

List of Figures

List of Tables

Acknowledgements

My first and foremost thanks go to my supervisor, Dr. Rabab Ward, for her supervision, friendly patience, academic guidance and generous support. This work would not be possible without her close attention. My special thanks also go to Dr. Saif Zahir for the technical support throughout this work.

I am grateful to Dr. Musleh Harbah and Dr. AbdulNasir Hussien for the guidance and the advice throughout my earlier years that make me what I am today. I am also thankful to my colleagues at the Digital Image Processing Lab specially our research engineer Dr. Alen Docef, and my friend Dr. Ismaeil Ismaeil.

Finally, I can not thank my parents enough, especially my mother, for the moral support and spiritual guidance, and my brothers and sisters for their patience and understanding.

Abdul Karim MURAD AGHA

The University of British Columbia

October 2000

To my beloved Mother Aicha

1 Introduction

1.0 Motivations

Expansion of color images is a vital process for printing images on large format printers especially for images with low resolution mode. Image expansion introduces many new pixels to the original image that will cause distortion and significant degradation in the quality of the image. Several methods have been employed to minimize such distortion: (1) Pixel replication; (2) Pixels averaging; (3) Edge detecting method and others. In this research we introduce a new segmentation-based technique that segment the image into regions. The boundaries of these regions i.e. the image edges are expanded so the resultant image has close resemblance with the original one. Such a method tends to retain the characteristics and features of the original image.

1.1 Digital Image Expansion: Background

Image expansion is a very important part of the digital image processing field. It is used in many applications such as, multimedia, image communication, digital photography, and medical imaging. Digital images are expanded by fitting an image with a continuous

model and resampling this model on a new expanded sampling grid. Most of image interpolation methods can be grouped under one of the four categories. These are 1) conventional interpolation methods, 2) adaptive methods, 3) statistical methods, and 4) new recent methods.

1.1.1 Interpolation Methods

Interpolation methods form the conventional methods in which an interpolation function is applied indiscriminately on the whole image. The first serious effort in image interpolation field is the sampling and interpolation discussion presented by Schafer et al. in 1973 [1]. The fastest, simplest, and most commonly used methods are: 1) The replication method, which also is called nearest neighbor or zero-degree interpolation. This method is equivalent to a sinc function in the frequency domain and is a poor low pass filter [2, 3]. 2) The bilinear (first-degree or linear) interpolation method; this method attenuates frequencies near the cutoff frequency, resulting in image blurring and smoothing [4, 5]. These two methods are the simplest and fastest methods, but they suffer from either block effects or blurring (over-smoothing effects). 3) Interpolation method with higher degrees such as quadratic (second-degree) or cubic (third-degree) interpolations these methods can produce better results than the zero or first degree interpolation methods. 4) Sinc interpolation, which has considerable energy over extended distance, and requires pruning [6, 7], or truncating [8, 9]. 5) Radial basis functions, which avoid blockiness effects [10]. 6) Cubic spline interpolation which tends to cause overshooting at sharp edges, producing ringing effects [11, 12]. 7) High resolution cubic splines (cubic convolution) [2, 3, 13, 14, 15, 16, 17], 8) Spline-like

methods; these methods simulate the spline function and produce similar results to those of the spline but use much simpler functions [15, 18, 19]. 9) Generalized spline filters based on partial differential equation (PDE) for image models [20, 21]. 10) IIR filtering [22] and Fourier transform where frequency interpolation is used [23, 24, 25, 26].

There are other interpolation functions such as Mitchmod function [27], and Mitchell and Smith [28]. Some of these conventional interpolation methods produce better results than others. These methods are not optimal, in the sense that they are not designed to minimize degradations and artifacts, and lack the consideration of the local characteristics of the image.

1.1.2 Adaptive Methods

Typically, an image with lower spatial resolution has relatively lower bandwidth in the Fourier domain. The frequency response of an interpolation function normally approximates an ideal low pass filter with the cutoff frequency equal to the Nyquist frequency of the sampled image. An interpolated image frequency response still lies within the original image bandwidth. Even an ideal interpolation method only preserves the original frequency components within the input image bandwidth. To expand a low-resolution image to high-resolution image with sharp edges, such an expansion algorithm should be able to generate high frequency components in the spectrum that correlate to the original frequency components. For that reason, a number of adaptive expansion methods were developed. An adaptive technique was introduced in the late 1980's by Wang et al. [29]. Other adaptive methods include an: adaptive interpolation method

based on the local structure of the image and which takes advantage of the intensity variations that are indistinguishable by the human eye [30, 31]. Lee et al., introduced a fast adaptive B-spline interpolation method. This method produces block or mosaic effects [32]. Nonlinear interpolation methods based on a source model and incorporating a local edge fitting were also developed [33, 34, 35, 36, 37]. Directional interpolation methods based on the local analysis of the spatial image structure were discussed in [17, 38, 39]. Enhancing edge during scaling [40, 41, 42, 43] and edge-restricted spatial interpolation based on cubic spline-under-tension kernel were also introduced [44].

1.1.3 Statistical Methods

Statistical methods usually involve and exploit some texture parameters. The methods in this category require that such parameters be estimated for the prescribed underlying image model. Parameters estimation is, however, a computationally demanding procedure. This limits the usefulness of these methods for any real time applications [35]. This category includes methods that are based on image models such as, Markov random fields [45, 46], Gibbs random field [47, 48, 1, 4], and others [49].

1.1.4 Other Methods

The fourth category of image expansion methods forms recent methods. This category is based on methods such as: DCT, neural networks; and wavelets. Image interpolation methods that use DCT coefficients of 8x8 blocks of the original image are introduced in [50, 51, 52, 53]. The use of Radial Basis Functions Network (RBFN) for image interpolation is also introduced [54, 55]. Random Neural Network (RNN) is also used to

interpolate images [56]. Other work uses Wavelet techniques to interpolate images [57, 58, 59].

2 Image Expansion Methods

2.0 Introduction

Many methods are developed for image expansion. Some are very popular and commonly used, some are theoretical, and others lie in between. The Replication and the Linear interpolation methods are very popular. The spline interpolation method is more theoretical than practical. In this chapter different methods for image expansion are discussed.

2.1 Replication Method

The pixel replication method simply maps each pixel in the original image onto an N×N block of pixels, where N is the expansion factor, and each of the pixels of the N × N block has the value of the original pixel. For example, for an expansion factor 3, each pixels in the original image is mapped onto a 3 × 3 block of pixels in the expanded image. Such a process is usually followed by some sort of a local correction heuristic that attempts to change the appropriate pixel gray levels depending on their neighborhood

in the original image [42]. This method is also known as zero-order interpolation. This method is very simple and cheap to implement, however, it introduces a tiling or "jaggies" effect which is known as a blockiness effect, specially at high expansion factors.

For N = 4, the normalized mapping process is as follows

$$\mathbf{1} \Rightarrow \begin{bmatrix} \mathbf{1} & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 \end{bmatrix}$$

2.1.1 One-Dimension Replication

Let the original data set be [4 1 3 2], as shown in **Figure 2.1**, and assume it is to be expanded 4 times.

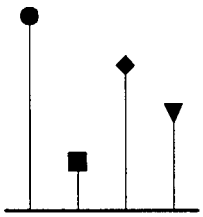


Figure 2.1 The original one dimension data

The expanded result is [**4** 4 4 4 **1** 1 1 1 **3** 3 3 3 **2** 2 2 2], the original values are bolded, **Figure 2.2** shows this process.

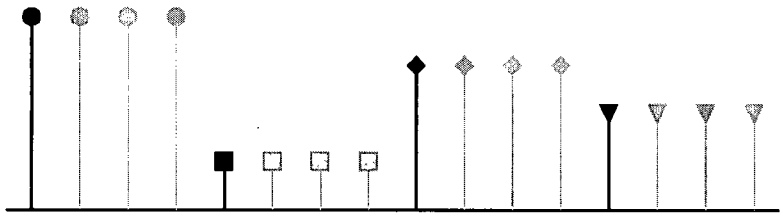


Figure 2.2 The result of pixel replication method (4 times)

2.1.2 Two-Dimension Replication

The original 2-dimentional set of values shown below is expanded 4 times. Each pixel is replicated 4 times in both the x and the y-axis directions, as follows

0	**20**	**40**	=>	**0**	0	0	0	**20**	20	20	20	**40**	40	40	40
20	**60**	**80**		0	0	0	0	20	20	20	20	40	40	40	40
40	**80**	**60**		0	0	0	0	20	20	20	20	40	40	40	40
				0	0	0	0	20	20	20	20	40	40	40	40
				20	20	20	20	**60**	60	60	60	**80**	80	80	80
				20	20	20	20	60	60	60	60	80	80	80	80
				20	20	20	20	60	60	60	60	80	80	80	80
				20	20	20	20	60	60	60	60	80	80	80	80
				40	40	40	40	**80**	80	80	80	**60**	60	60	60
				40	40	40	40	80	80	80	80	60	60	60	60
				40	40	40	40	80	80	80	80	60	60	60	60
				40	40	40	40	80	80	80	80	60	60	60	60

2.1.3 Algorithms

The pixel replication algorithm is separable. That means it can be applied as a row one-dimension pixel replication followed by a column one-dimension pixel replication.

2.2 Linear Interpolation Method

This method finds an estimate for an unknown point by linearly interpolating the values of its neighboring points. For the one-dimensional case linear interpolation finds the estimate of the unknown point from its 2 nearest points whose values are known. This involves fitting a straight line through the two points. This method is also called first-order interpolation or pixel averaging. Linear interpolation is the easiest interpolation method. In the two-dimension case, the linear interpolation method is called bilinear interpolation and involves finding the unknown value of a point from four known points. The bilinear interpolation method is relatively simple, fast and computationally inexpensive. On the other hand, it introduces blurring and contouring effects around edges in the expanded image.

2.2.1 Mathematical Representation

In one dimension case, given two data points (pairs of numbers) (x_0, f_0), (x_1, f_1), linear interpolation finds a polynomial $p_1(x)$ such that $p_1(x_0) = f_0$, and $p_1(x_1) = f_1$. Now $p_1(x)$ can be written as [60]

$$p_1(x) = f_0 + (x - x_0)\, f[x_0, x_1] \qquad (1)$$

where $f[x_0, x_1]$ is the first divided difference and is given by $\frac{f_1 - f_0}{x_1 - x_0}$. Assuming the distance between x_1 and $x_0 = 1$, i.e. $x_1 - x_0 = 1$ then $f[x_0, x_1] = f_1 - f_0$.

The subscript 1 of $p_1(x)$ indicates the degree of the polynomial, which is 1 for the linear interpolation polynomial. Equation (1) can now be written as

$$p_1(x) = f_0 + (x - x_0)\,(f_1 - f_0). \qquad (2)$$

From that we can see that $p_1(x_0) = f_0$, $p_1(x_1) = f_1$. Let $r = (x - x_0)$ then

$$\begin{aligned} p_1(x) = p_1(x_0 + r) \quad &= f_0 + r\,(f_1 - f_0) \\ &= f_0 + r\ f_1 - r\ f_0 \\ &= (1 - r)\ f_0 + r\ f_1 \qquad (3) \end{aligned}$$

i.e. the interpolated value $p_1(x)$ is a weighted sum of f_0 and f_1 where f_0 and f_1 contribute $(1 - r)f_0$ and $r\ f_1$, respectively, towards the value at a point x as shown in **Figure 2.3**.

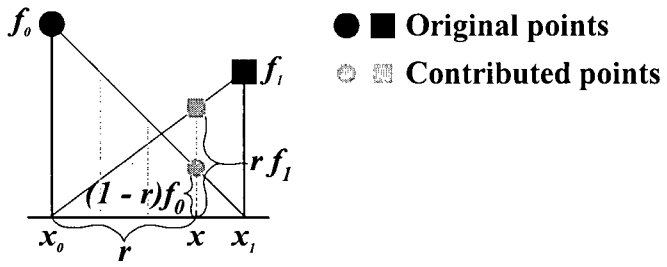


Figure 2.3 The linearly contributed values from the two neighbor points

In other word, f_0 is mapped onto $\{ (1 - r)\ f_0 \}$ to its right, and f_1 is mapped onto $\{ r\ f_1 \}$ to its left. And the expanded image will be the result of adding the overlapped mapped values. This expansion could be easily shown to be equivalent to convolving the input image with a system whose impulse response function for an expansion factor N = 4 is as shown in **Figure 2.4**.

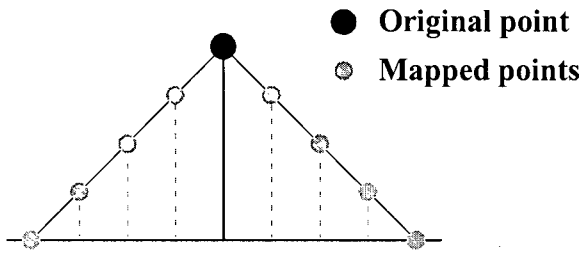


Figure 2.4 The original value, and its mapped result, in the 4 times expansion case.

In the two-dimensional case, four data points are given (x_0, y_0, f_0), (x_1, y_1, f_1), (x_2, y_2, f_2), and (x_3, y_3, f_3). The bilinear interpolation is three linear (one-dimensional) interpolations.

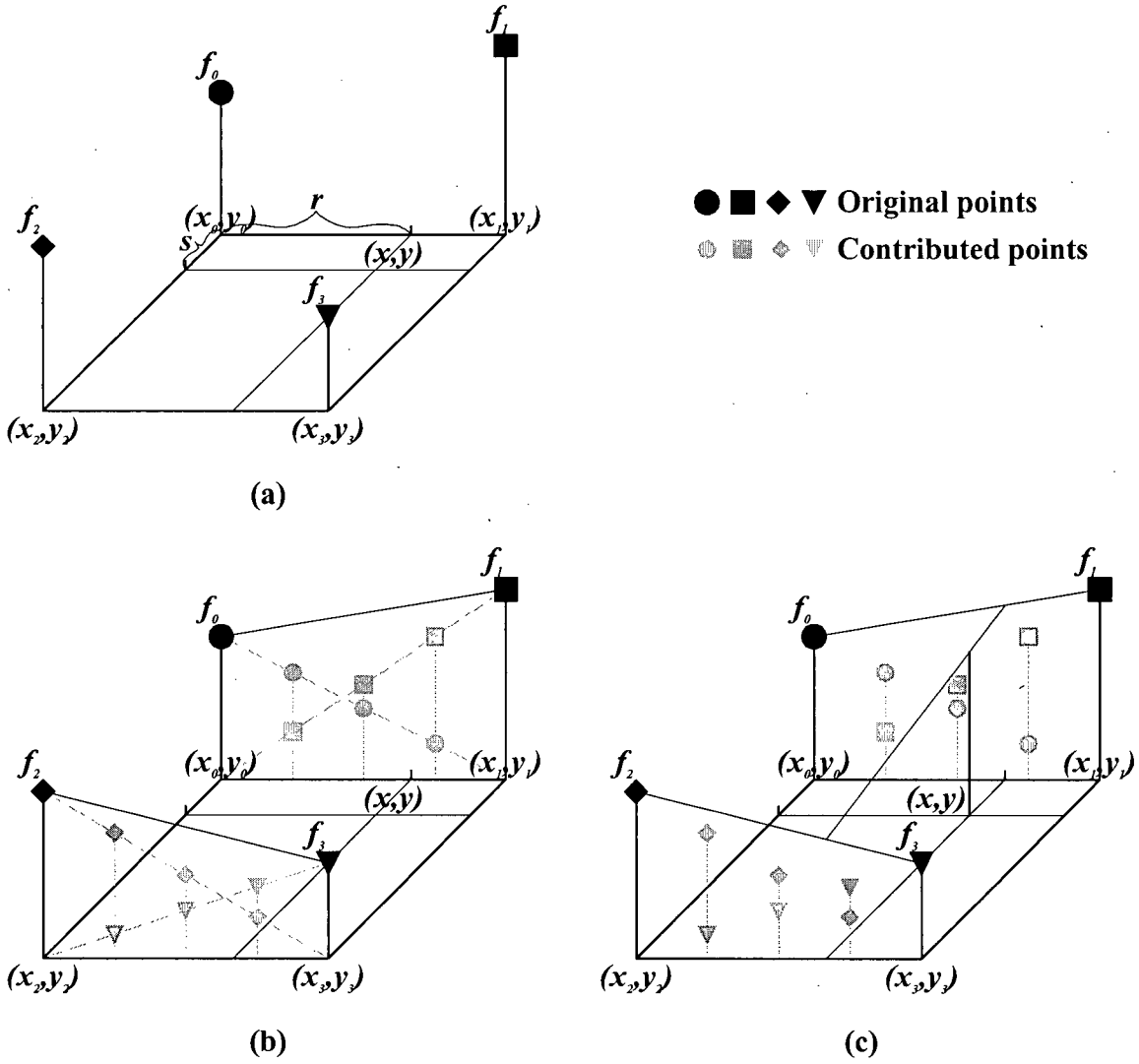


Figure 2.5 Bilinear Interpolation process.

Similar to the assumptions in the one-dimensional case, let $r = (x - x_0)$ and $s = (y - y_0)$ as shown in **Figure 2.5**.

The first linear interpolation is a horizontal interpolation between f_0 and f_1 and finds a polynomial $p_{11}(x)$, such that $p_{11}(x_0) = f_0$, and $p_{11}(x_1) = f_1$. Using (3) where $r = (x - x_0)$, $p_{11}(x)$ can be written as

$$p_{11}(x_0 + r) = (1 - r)\, f_0 + r\, f_1 \qquad (4)$$

The second linear interpolation is a horizontal interpolation between f_2 and f_3 and finds a polynomial $p_{12}(x)$, such that $p_{12}(x_2) = f_2$, and $p_{12}(x_3) = f_3$. $r = (x - x_2)$ then $p_{12}(x)$ can be written as

$$p_{12}(x_2 + r) = (1 - r)\, f_2 + r\, f_3 \qquad (5)$$

Now consider a point $(x_0 + r,\ y_0 + s)$, the third linear interpolation is a vertical interpolation and finds a polynomial $p_1(x, y)$ that passes through $p_{11}(x_0 + r)$, and $p_{12}(x_2 + r)$. Now $p_1(x, y)$ can be written as

$$\begin{aligned} p_1(x_0 + r, y_0 + s) &= (1 - s)\, p_{11}(x_0 + r) + (s)\, p_{12}(x_2 + r) \\ &= (1 - s)(1 - r)\, f_0 + (1 - s)(r)\, f_1 + (s)(1 - r)\, f_2 + (s)(r)\, f_3 \qquad (6) \end{aligned}$$

i.e. the interpolated value $p_1(x, y)$ is a weighted sum of f_0, f_1, f_2, and f_3.

In other word, the pixel (x_0, y_0) contributes $\{ (1 - s)(1 - r) f_0 \}$ to its lower-right pixels, the pixel (x_1, y_1) contributes $\{ (1 - s)(r) f_1 \}$ to its lower-left pixels, the pixel (x_2, y_2) contributes $\{ (s)(1 - r) f_2 \}$ to its upper-right pixels, and the pixel (x_3, y_3) contributes $\{ (s)(r) f_3 \}$ to its upper-left pixels. And the expanded image will be the result of adding these four contributions. This can be shown to be equivalent to convolving the original image with a system whose impulse response function e.g. for the case of expansion factor N = 4 is as shown in **Figure 2.6**.

$$\mathbf{1} \quad \Rightarrow \quad \frac{1}{16} \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 2 & 3 & 4 & 3 & 2 & 1 & 0 \\ 0 & 2 & 4 & 6 & 8 & 6 & 4 & 2 & 0 \\ 0 & 3 & 6 & 9 & 12 & 9 & 6 & 3 & 0 \\ 0 & 4 & 8 & 12 & \mathbf{16} & 12 & 8 & 4 & 0 \\ 0 & 3 & 6 & 9 & 12 & 9 & 6 & 3 & 0 \\ 0 & 2 & 4 & 6 & 8 & 6 & 4 & 2 & 0 \\ 0 & 1 & 2 & 3 & 4 & 3 & 2 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

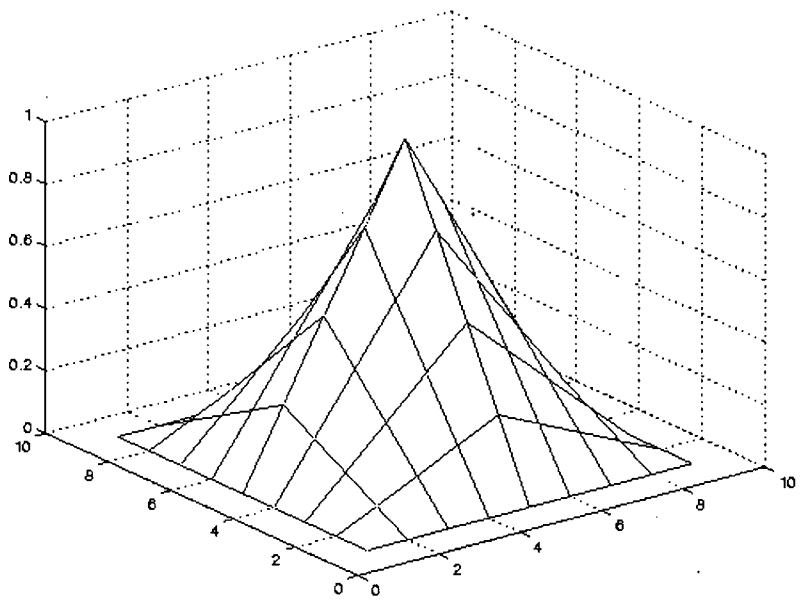


Figure 2.6 The model (spread unction) of Bilinear Interpolation

2.2.2 One-Dimension Interpolation

Let the original data set be [4 1 3 2], as shown in **Figure 2.7**, and assume it is to be expanded 4 times.

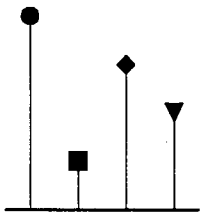
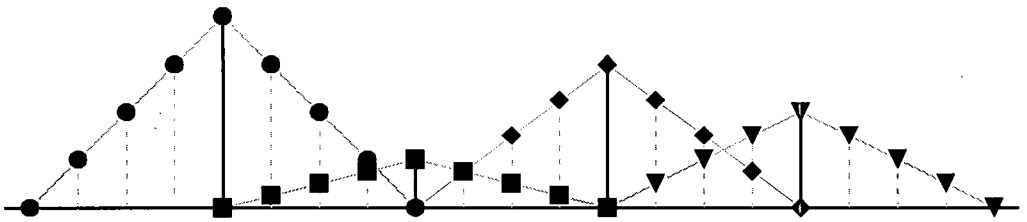


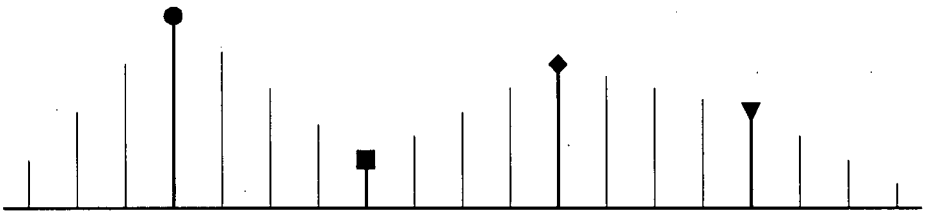
Figure 2.7 The original one dimension data

Zero padding is first used to complete the process. This is followed by mapping these data onto the expanded domain, then adding the overlapped values. The result of expansion to the right direction is [**4** 3.25 2.5 1.75 **1** 1.5 2 2.5 **3** 2.75 2.5 2.25 **2** 1.5 1 0.5

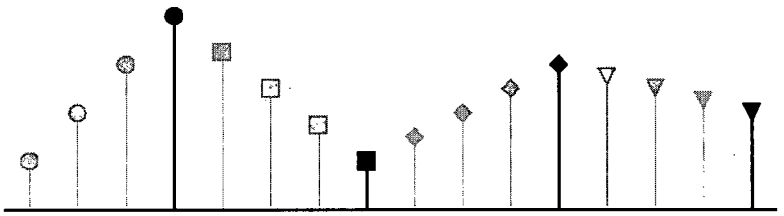
]. And the result of expansion to the left direction is [1 2 3 **4** 3.25 2.5 1.75 **1** 1.5 2 2.5 **3** 2.75 2.5 2.25 **2**], the original data is bolded for convenience, **Figure 2.8** shows this process.



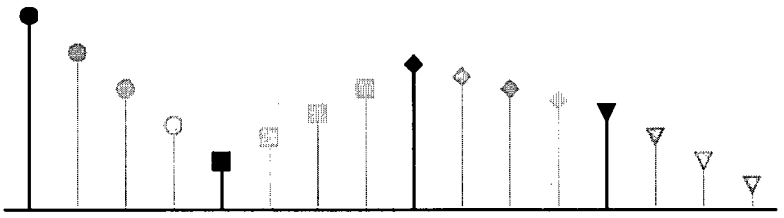
(a)



(b)



(c)



(d)

Figure 2.8 (a) The mapped data onto the expanded domain, (b) the result of adding the overlapped values, (c) the expansion result to the right, (d) the expansion result to the left.

2.2.3 Two-Dimension Interpolation

Assume the original 2-dimentional image is to be expanded 4 times, so that each pixel is mapped onto a 7x7 block then the overlapped values are added, then rounded to the nearest integer to produce the final expanded result.

0	**20**	**40**
20	**60**	**80**
40	**80**	**60**

=>

0	5	10	15	**20**	25	30	35	**40**	30	20	10
5	11	18	24	30	35	40	45	50	38	25	13
10	18	25	33	40	45	50	55	60	45	30	15
15	24	33	41	50	55	60	65	70	53	35	18
20	30	40	50	**60**	65	70	75	**80**	60	40	20
25	35	45	55	65	68	70	73	75	56	38	19
30	40	50	60	70	70	70	70	70	53	35	18
35	45	55	65	75	73	70	68	65	49	33	16
40	50	60	70	**80**	75	70	65	**60**	45	30	15
30	38	45	53	60	56	53	49	45	34	23	11
20	25	30	35	40	38	35	33	30	23	15	8
10	13	15	18	20	19	18	16	15	11	8	4

2.2.4 The Algorithm

Bilinear interpolation algorithm is a separable one. That means it can be applied as a row one-dimension linear interpolation on every row followed by a column one-dimensional linear interpolation on every column.

2.3 Quadratic Interpolation Method

Quadratic interpolation uses a second-degree polynomial whose curve passes through three points. Using three points f_0, f_1, and f_2 to find interpolated values for points between f_0 and f_1 is called forward interpolation, and to interpolate values for points between f_1 and f_2 is called backward interpolation [60]. In this section, only forward quadratic interpolation is considered. For the two-dimensional case it is called biquadratic interpolation, and it involves nine pixels. That means better quality than linear interpolation. But it still introduces blurry results.

2.3.1 Mathematical Representation

In the one-dimension case, given three data points (three pairs of numbers) (x_0, f_0), (x_1, f_1), (x_2, f_2), quadratic interpolation finds a polynomial $p_2(x)$ such that $p_2(x_0) = f_0$, $p_2(x_1) = f_1$, and $p_2(x_2) = f_2$. The quadratic polynomial is [60, 61]

$$p_2(x) = f_0 + (x - x_0)\, f[x_0, x_1] + (x - x_0)(x - x_1)\, f[x_0, x_1, x_2] \qquad (7)$$

where the subscript 2 of $p_2(x)$ indicates the degree of the polynomial. $f[x_0, x_1]$ is the first divided difference and is given by $\frac{f_1 - f_0}{x_1 - x_0} = f_1 - f_0$, where $x_1 - x_0 = 1$, and $f[x_0, x_1, x_2]$ is the second divided difference and given by $\frac{f[x_1, x_2] - f[x_0, x_1]}{x_2 - x_0} = 0.5\,(f_2 - 2\,f_1 + f_0)$.

From the above we can deduce that $p_2(x_0) = f_0$, $p_2(x_1) = f_1$, $p_2(x_2) = f_2$. Let $r = (x - x_0)$,

then $p_2(r) = f_0 + r\,(f_1 - f_0) + 0.5\,r\,(r-1)\,(f_2 - 2\,f_1 + f_0)$

$$= 0.5\,(\,(r-1)\,(r-2)\,f_0 + 2\,r\,(2-r)\,f_1 + r\,(r-1)\,f_2\,) \qquad (8)$$

To find the interpolated value for an unknown point lying between f_0 and f_1, we notice from (8) that f_0 contributes $\{\ 0.5\ (r-1)(r-2)\ f_0\ \}$, f_1 contributes $\{\ (r)(2-r)\ f_1\ \}$, and f_2 contributes $\{\ 0.5\ (r)(r-1)\ f_2\ \}$, to this point. Thus for an expansion factor = 4 it can be shown that each original point will be mapped onto eleven values in the expanded result as shown in **Figure 2.9**. After mapping all the original points, the expanded data will be the result of adding the overlapped mapped values.

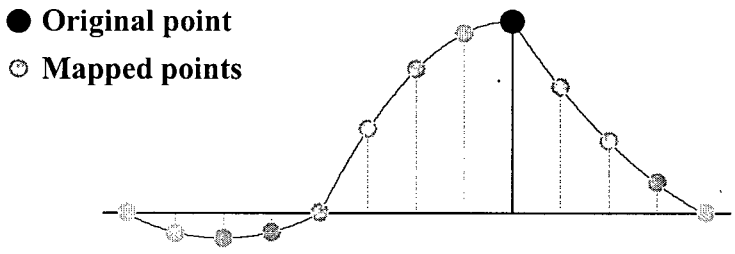


Figure 2.9 The original value, and its mapped result, in 4-time expansion case.

The two-dimensional mapping function can be found by simply convolving a horizontal one-dimensional mapping function with a vertical one-dimensional mapping function. As an example consider an expansion factor 4. A pixel in the original image is mapped onto an 11×11 block in the expanded image according to **Figure 2.10**.

$$\mathbf{1} \quad \Rightarrow \quad \frac{1}{64} \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & .6 & .8 & .6 & 0 & -3 & -5 & -6 & -6 & -4 & -2 & -1 & 0 \\ 0 & .8 & 1 & .8 & 0 & -4 & -6 & -8 & -8 & -5 & -3 & -1 & 0 \\ 0 & .6 & .8 & .7 & 0 & -3 & -5 & -6 & -6 & -4 & -2 & -1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & -3 & -4 & -3 & 0 & 12 & 21 & 26 & 28 & 18 & 11 & 4 & 0 \\ 0 & -5 & -6 & -5 & 0 & 21 & 36 & 45 & 48 & 32 & 18 & 8 & 0 \\ 0 & -6 & -8 & -6 & 0 & 26 & 45 & 56 & 60 & 39 & 23 & 9 & 0 \\ 0 & 6 & -8 & -6 & 0 & 28 & 48 & 60 & \mathbf{64} & 42 & 24 & 10 & 0 \\ 0 & -4 & -5 & -4 & 0 & 18 & 31 & 39 & 42 & 28 & 16 & 7 & 0 \\ 0 & -2 & -3 & -2 & 0 & 11 & 18 & 23 & 24 & 16 & 9 & 4 & 0 \\ 0 & -1 & -1 & -1 & 0 & 4 & 8 & 9 & 10 & 7 & 4 & 2 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

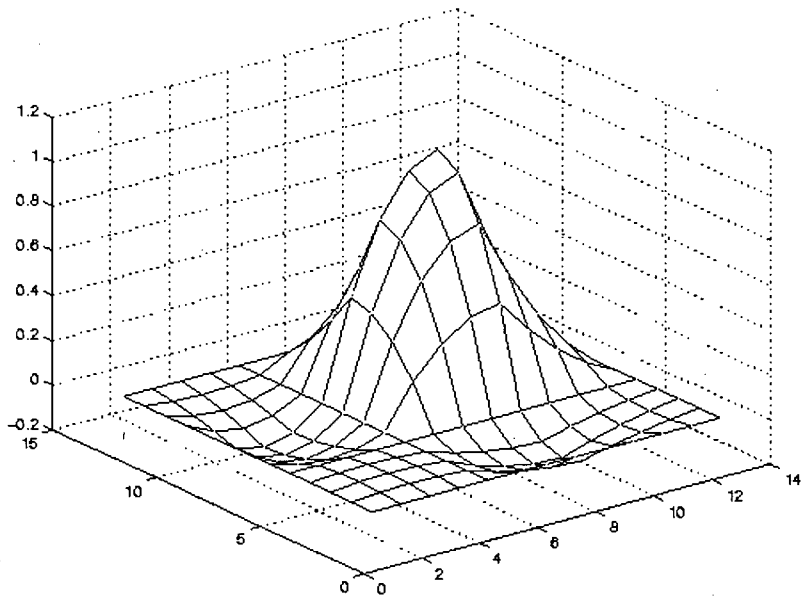


Figure 2.10 The model (spread function) of Biquadratic Interpolation

2.3.2 One-dimension Interpolation

Let the original data set be [4 1 3 2], as shown in **Figure 2.11**, and assume it is to be expanded 4 times.

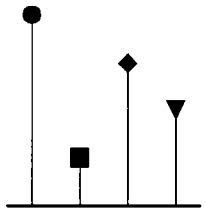
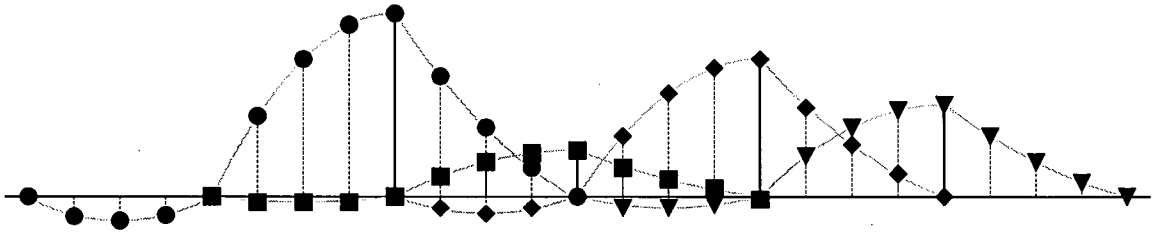
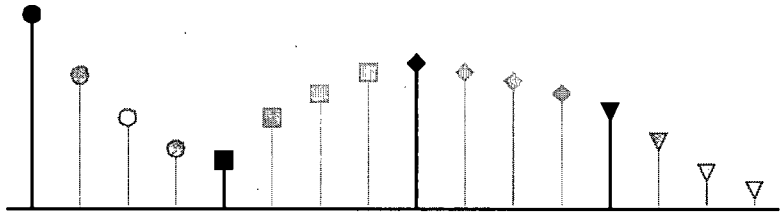


Figure 2.11 The original one dimension data

Mapping each datum onto the expanded domain, then adding the overlapped values, the expanded result to the right direction is [**4** 2.78 1.88 1.28 **1** 1.97 2.63 2.97 **3** 1.97 1.13 0.47 **2** 1.31 0.75 0.31], **Figure 2.12** shows this process.



(a)



(b)

Figure 2.12 (a) The mapped data onto the expanded domain, (b) The expansion result to the right direction.

2.3.3 Two-Dimension Interpolation

Assume the original 2-dimentional image is to be expanded 4 times. In this case each pixel is mapped onto an 11x11 block then the overlapped values are added, then rounded to the nearest integer. The following is the expansion result to the right direction.

0	**20**	**40**	
20	**60**	**80**	=>
40	**80**	**60**	

0	5	10	15	**20**	31	38	41	**40**	26	15	6
5	12	19	25	32	45	54	57	56	37	21	9
10	19	27	35	43	57	67	70	68	44	25	11
15	25	35	44	52	67	76	79	76	50	28	12
20	32	43	52	**60**	74	82	84	**80**	53	30	13
31	45	57	67	74	83	87	85	79	52	30	12
38	54	67	76	83	87	87	83	75	49	28	12
41	57	70	79	84	85	83	78	69	45	26	11
40	56	68	76	**80**	79	75	69	**60**	39	23	9
26	37	44	50	53	52	49	45	39	26	15	6
15	21	25	28	30	30	28	26	23	15	08	4
6	9	11	12	13	12	12	11	9	6	4	1

2.3.4 Algorithm

Biquadratic interpolation algorithm is a separable one. That means it can be applied as a one-dimension quadratic interpolation on every row and then as a one-dimensional quadratic interpolation on every column.

2.4 Cubic Interpolation

Cubic interpolation is the process of interpolation using third order degree polynomials interpolation using the four points (pixels) x_{-1}, x_0, x_1 and x_2, to interpolate values lying between the first and second points x_{-1} and x_0 is called forward cubic interpolation. Interpolation of points lying between the third and fourth points using the 4 known points is called backward cubic interpolation. The most common interpolation is the central one and involves interpolation between the second x_0 and the third points x_1 [60, 61]. In this section, only the central cubic interpolation is considered. For the two-dimensional case it is called bicubic interpolation, and it uses sixteen pixels. Interpolating values lying amongst the four points at the center of the sixteen known pixels is called central bicubic interpolation and is the process used here. Although this method produces sharper transition areas at edges, it still introduces blurry results and oscillations at the edges and is computationally expensive. This method produces better quality than the previous discussed interpolation methods.

2.4.1 Mathematical Representation

In the one-dimension case, given four data points (x_{-1}, f_{-1}), (x_0, f_0), (x_1, f_1), and (x_2, f_2), cubic interpolation finds a polynomial $p_3(x)$ such that $p_3(x_{-1}) = f_{-1}$, $p_3(x_0) = f_0$, $p_3(x_1) = f_1$, and $p_3(x_2) = f_2$. The cubic polynomial formula is

$$p_3(x) = f_0 + (x - x_0)\, f[x_0, x_1] + (x - x_0)(x - x_1)\, f[x_0, x_1, x_2]$$

$$+ (x - x_0)(x - x_1)(x - x_2)\, f[x_{-1}, x_0, x_1, x_2] \qquad (9)$$

where the subscript 3 of $p_3(x)$ indicates the degree of the polynomial [60, 61].

$f[x_0, x_1]$ is the first divided difference and is given by

$$\frac{f_1 - f_0}{x_1 - x_0} = f_1 - f_0., \text{ when } x_1 - x_0 = 1.$$

$f[x_0, x_1, x_2]$ is the second divided difference and is given by

$$\frac{f[x_1, x_2] - f[x_0, x_1]}{x_2 - x_0} = 0.5\,(f_2 - 2\,f_1 + f_0), \text{ and}$$

$f[x_0, x_1, x_2, x_3]$ is the third divided difference and is given by

$$\frac{f[x_0, x_1, x_2] - f[x_{-1}, x_0, x_1]}{x_2 - x_{-1}} = \frac{0.5(f_2 - 2f_1 + f_0) - 0.5(f_1 - 2f_0 + f_{-1})}{3}.$$

Let $r = (x - x_0)$, so

$$p_3(r) = f_0 + r\,(f_1 - f_0) + 0.5\,r\,(r-1)\,(f_2 - 2\,f_1 + f_0) + r\,(r-1)\,(r-2)\,(f_2 - 3\,f_1 - 3\,f_0 + f_{-1})$$

$$= \frac{1}{6}\,[\,(-\,3\,r\,(r-1)^2\,f_{-1} + 3\,(\,r^2\,(3\,r-5) + 2\,)\,f_0 - r\,(3\,r^2 - 4\,r - 1)\,f_1 + 3\,r^2\,(r-1)\,f_2\,)\,] \qquad (10)$$

For a point lying between x_0 and x_1, f_{-1} contributes $\{ -0.5\,(r)(r-1)^2\,f_{-1} \}$ towards its interpolated value, f_0 contributes $\{ 0.5\,((r)^2(3\,r-5) + 2)\,f_0 \}$, f_1 contributes $\{ -0.5\,(r)(3r^2 - 4r - 1)\,f_1 \}$, and f_2 contributes $\{ 0.5\,(r)^2(r-1)\,f_2 \}$. For an example, consider the case when the expansion factor is 4. It can be shown that the value of an original pixel will be mapped onto fifteen values in the expanded result as shown in **Figure 2.13**. After mapping all the original points, the expanded data will be the result of adding the overlapped mapped values.

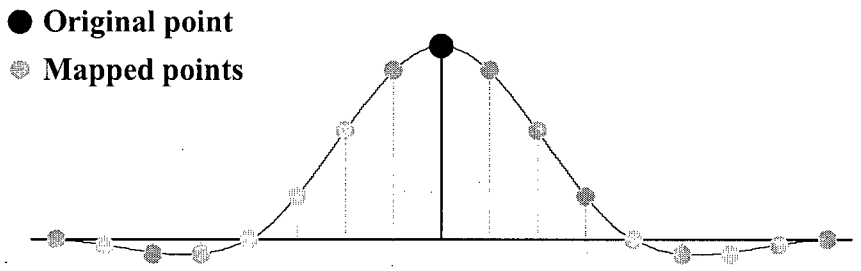


Figure 2.13 The original value, and its mapped result, in 4-time expansion case.

For the two-dimensional mapping function, it can be shown that this could be accomplished by simply convolving a horizontal one-dimensional mapping function with a vertical one-dimensional mapping function. As an example consider an expansion factor 4. A pixel in the original image is mapped onto a 15 × 15 block in the expanded image as shown in **Figure 2.14**.

$$1 \quad \Rightarrow \quad \frac{1}{64}\begin{bmatrix}
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & .1 & .1 & 0 & -.4 & -.8 & -1 & -2 & -1 & -.8 & -.4 & 0 & .1 & .1 & 0 & 0 \\
0 & .1 & .3 & .3 & 0 & -1 & -2 & -4 & -4 & -4 & -2 & -1 & 0 & .3 & .3 & .1 & 0 \\
0 & .1 & .3 & .3 & 0 & -1 & -3 & -4 & -5 & -4 & -3 & -1 & 0 & .3 & .3 & .1 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & -.3 & -1 & -1 & 0 & 3 & 8 & 13 & 15 & 13 & 8 & 3 & 0 & -1 & -1 & -.3 & 0 \\
0 & -.8 & -2 & -3 & 0 & 8 & 20 & 31 & 36 & 31 & 20 & 8 & 0 & -3 & -2 & -.8 & 0 \\
0 & -1 & -4 & -4 & 0 & 13 & 31 & 48 & 56 & 48 & 31 & 13 & 0 & -4 & -4 & -1 & 0 \\
0 & -2 & -4 & -5 & 0 & 15 & 36 & 56 & \mathbf{64} & 56 & 36 & 15 & 0 & -5 & -4 & -2 & 0 \\
0 & -1 & -4 & -4 & 0 & 13 & 31 & 48 & 56 & 48 & 31 & 13 & 0 & -4 & -4 & -1 & 0 \\
0 & -.8 & -2 & -3 & 0 & 8 & 20 & 31 & 36 & 31 & 20 & 8 & 0 & -3 & -2 & -.8 & 0 \\
0 & -.3 & -1 & -1 & 0 & 3 & 8 & 13 & 15 & 13 & 8 & 3 & 0 & -1 & -1 & -.3 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & .1 & .3 & .3 & 0 & -1 & -3 & -4 & -5 & -4 & -3 & -1 & 0 & .3 & .3 & .1 & 0 \\
0 & .1 & .3 & .3 & 0 & -1 & -2 & -4 & -4 & -4 & -2 & -1 & 0 & .3 & .3 & .1 & 0 \\
0 & 0 & .1 & .1 & 0 & -.4 & -.8 & -1 & -2 & -1 & -.8 & -.4 & 0 & .1 & .1 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0
\end{bmatrix}$$

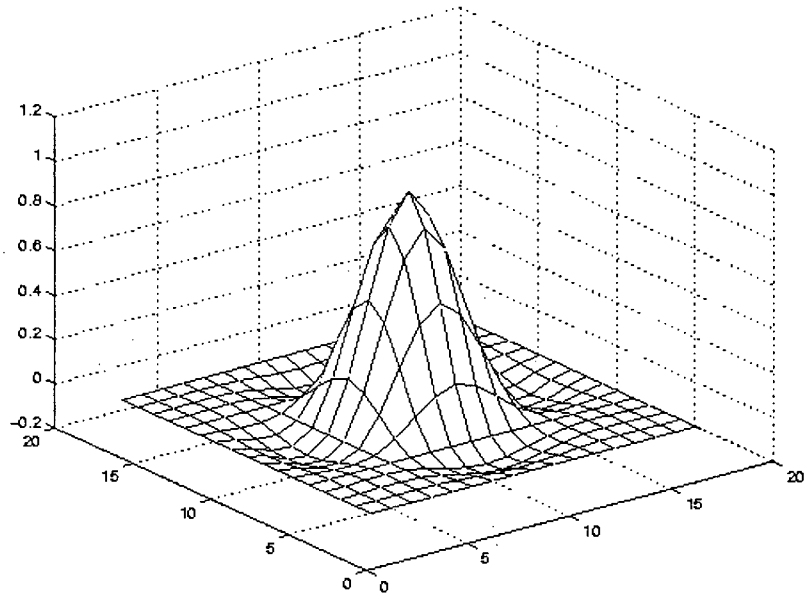


Figure 2.14 The model (spread function) of Bicubic Interpolation

2.4.2 One-dimension Interpolation

Let the original data set be [4 1 3 2], as shown in **Figure 2.15**, and assume it is to be expanded 4 times.

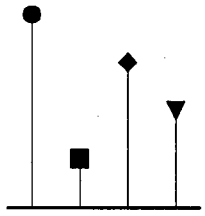


Figure 2.15 The original one dimension data

Mapping these data onto the expanded domain, then adding the overlapped values, the result of expansion to the right direction is [4 3.63 2.63 1.56 1 1.22 1.88 2.59 3 2.98 2.75 2.39 2 1.52 0.94 0.38], **Figure 2.16** shows this process.

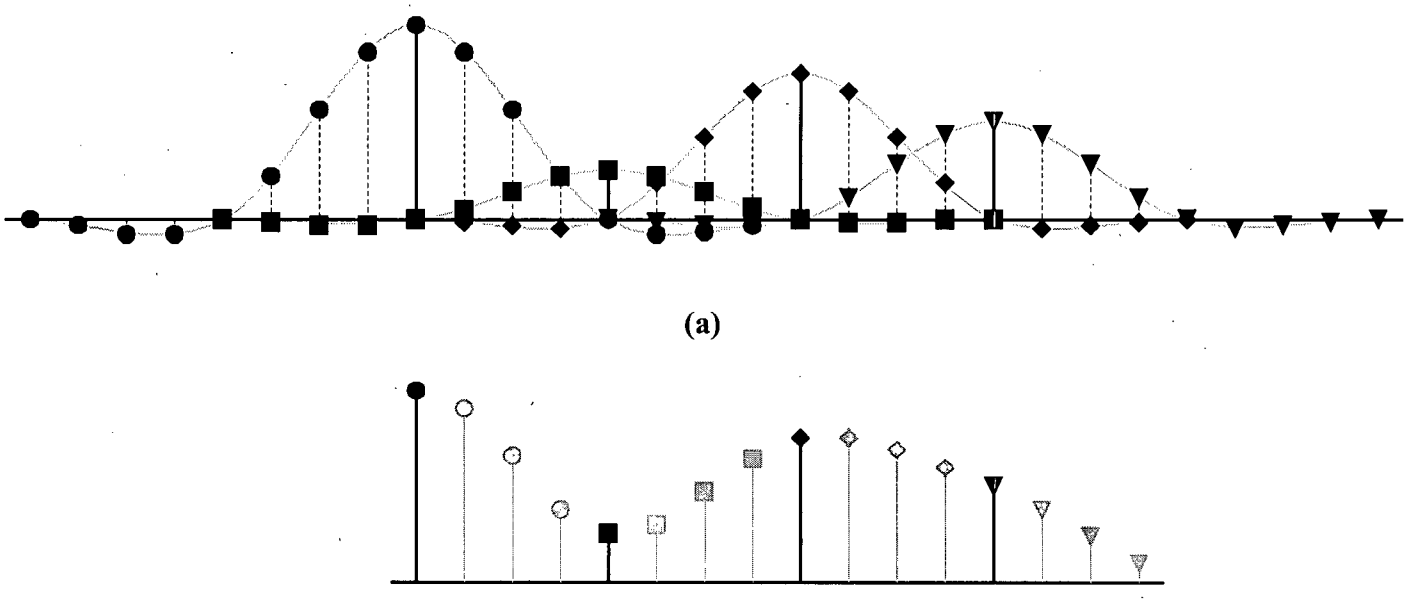


Figure 2.16 (a) The mapped data onto the expanded domain, (b) the result of adding the overlapped values

2.4.3 Two-Dimension Interpolation

Assume the original 2-dimentional image is to be expanded 4 times. In this case each pixel is mapped onto a 15x15 block then the overlapped values are added, then rounded to the nearest integer. The following is the result of expansion in the right direction.

0	**20**	**40**	**20**
20	**60**	**80**	**40**
40	**80**	**60**	**20**
20	**40**	**20**	**80**

=>

[0]	4	9	15	[20]	26	34	39	[40]	33	21	9	[20]	18	9	3
4	8	15	22	29	37	45	51	51	43	27	11	23	20	10	4
9	15	23	32	40	49	58	64	64	52	33	14	31	27	12	5
15	22	32	42	51	60	70	76	74	61	39	16	37	32	14	6
[20]	29	40	51	[60]	69	78	83	[80]	65	41	17	[40]	36	16	6
26	37	49	60	69	76	82	84	80	65	41	17	36	31	14	5
34	45	58	70	78	82	84	83	76	61	38	15	28	25	12	4
39	51	64	76	83	84	83	78	69	54	34	14	22	20	9	3
[40]	51	64	74	[80]	80	76	69	[60]	46	29	12	[20]	18	9	3
33	43	52	61	65	65	61	54	46	36	22	9	32	20	10	4
21	27	33	39	41	41	38	34	29	22	14	6	52	36	14	6
9	11	14	16	17	17	15	14	12	9	6	2	71	59	20	8
[20]	23	31	37	[40]	36	28	22	[20]	32	52	71	[80]	69	24	10
18	20	27	32	36	31	25	19	18	20	36	59	69	29	13	6
9	10	12	14	16	14	12	9	9	10	14	20	24	13	7	3
3	4	5	6	6	5	4	3	3	4	6	8	10	6	3	1

2.4.4 Algorithm

Bicubic interpolation process is a separable one. That means it can be applied as a one-dimensional cubic interpolation on every row and then as a one-dimensional cubic interpolation on every column.

2.5 Bézier Interpolation Method

The most popular Bézier curve is the Bézier cubic curve, which is a cubic polynomial curve segment, named after Pierre Bézier who developed these curves for use in designing automobiles at Renault. These curves are obtained by using the end points tangent vectors and the values of two intermediate points.

Bézier cubic interpolation involves 4 pixels in the one-dimensional case. The Bézier cubic curve passes through the two end points and finds an interpolated value for the other two-middle points. The cubic Bézier interpolation method simply fits a curve that starts at the first point, ends at the last one, and does not necessarily pass through the middle ones [28, 62]. This method is found to eliminate the blockiness effect at the edges, but it introduces unacceptable smoothness and blurring.

2.5.1 Mathematical Representation

Using four given data (x_0, f_0), (x_1, f_1), (x_2, f_2), and (x_3, f_3) the cubic Bézier interpolation finds a cubic polynomial p(x) given by [28, 62, 63, 64]

$$p(x) = \sum_{k=0}^{L} p_k B_k^L(x) \qquad (11)$$

where L is the order of the polynomial, p_k is the polynomial coefficients and the functions $B_k^L(x)$ are known as Bernstein polynomials, and are given by

$$B_k^L(x) = \binom{L}{k}(1-x)^{L-k}x^k \qquad (12)$$

$$\binom{L}{k} = \frac{L!}{k!(L-k)!}$$

So the Bézier curve between these four data points is given by

$$p(x) = \sum_{k=0}^{L} p_k * \frac{L!}{k!(L-k)!} * (1-x)^{L-k} x^k \qquad (13)$$

The expanded image at any point x will be the result of adding the overlapped mapped values of the weight contributions of the 4 nearest original points. For the case of N = 4 i.e the 4 times expansion, the contributions of the original value will spread (get mapped) onto fifteen values in the expanded result, which is the mapping function shown in **Figure 2.17**.

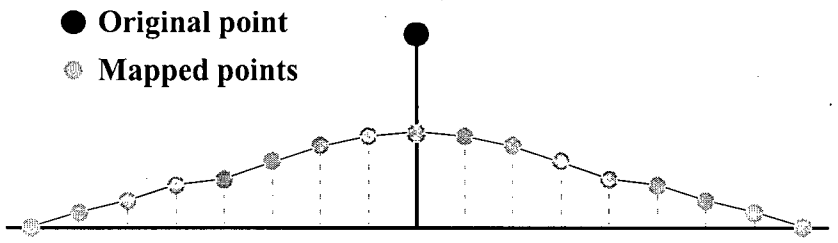


Figure 2.17 The original value, and its mapped result, in 4-time expansion case.

The two-dimensional mapping function can be found by simply convolving a horizontal one-dimensional mapping function with a vertical one-dimensional mapping function. As an example consider an expansion factor 4. A pixel in the original image is mapped onto a 15 × 15 block in the expanded image as shown in **Figure 2.18**.

$$\mathbf{1} \Rightarrow \frac{1}{24}\begin{bmatrix}
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 1 & 1 & 1 & 1 & 2 & 2 & 2 & 2 & 2 & 1 & 1 & 1 & 1 & 0 & 0 \\
0 & 1 & 1 & 2 & 2 & 2 & 3 & 3 & 4 & 3 & 3 & 2 & 2 & 2 & 1 & 1 & 0 \\
0 & 1 & 2 & 3 & 3 & 3 & 5 & 5 & 6 & 5 & 5 & 4 & 3 & 3 & 2 & 1 & 0 \\
0 & 1 & 2 & 3 & 3 & 4 & 5 & 6 & 6 & 6 & 5 & 4 & 3 & 3 & 2 & 1 & 0 \\
0 & 1 & 2 & 4 & 4 & 6 & 7 & 8 & 9 & 8 & 7 & 6 & 4 & 4 & 2 & 1 & 0 \\
0 & 2 & 3 & 5 & 5 & 7 & 9 & 10 & 11 & 10 & 9 & 7 & 5 & 5 & 3 & 2 & 0 \\
0 & 2 & 3 & 5 & 6 & 8 & 10 & 12 & 12 & 12 & 10 & 8 & 6 & 5 & 3 & 2 & 0 \\
0 & 2 & 4 & 6 & 6 & 9 & 11 & 12 & \mathbf{13} & 12 & 11 & 9 & 6 & 6 & 4 & 2 & 0 \\
0 & 2 & 3 & 5 & 6 & 8 & 10 & 12 & 12 & 12 & 10 & 8 & 6 & 5 & 3 & 2 & 0 \\
0 & 2 & 3 & 5 & 5 & 7 & 9 & 10 & 11 & 10 & 9 & 7 & 5 & 5 & 3 & 2 & 0 \\
0 & 1 & 2 & 4 & 4 & 6 & 7 & 8 & 9 & 8 & 7 & 6 & 4 & 4 & 2 & 1 & 0 \\
0 & 1 & 2 & 3 & 3 & 4 & 5 & 6 & 6 & 6 & 5 & 4 & 3 & 3 & 2 & 1 & 0 \\
0 & 1 & 2 & 3 & 3 & 4 & 5 & 5 & 6 & 5 & 5 & 4 & 3 & 3 & 2 & 1 & 0 \\
0 & 1 & 1 & 2 & 2 & 2 & 3 & 3 & 4 & 3 & 3 & 2 & 2 & 2 & 1 & 1 & 0 \\
0 & 0 & 1 & 1 & 1 & 1 & 2 & 2 & 2 & 2 & 2 & 1 & 1 & 1 & 1 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0
\end{bmatrix}$$

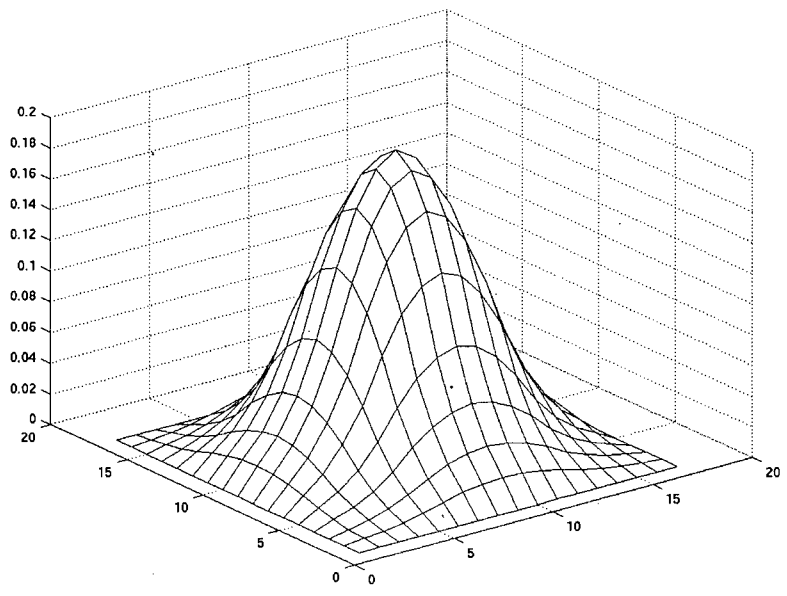


Figure 2.18 The model (spread function) of Bézier Interpolation

2.5.2 One-dimension Interpolation

Let the original data set be [4 1 3 2], as shown in **Figure 2.19**, and assume it is to be expanded 4 times.

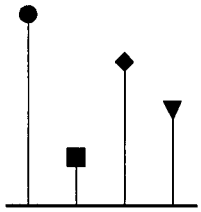
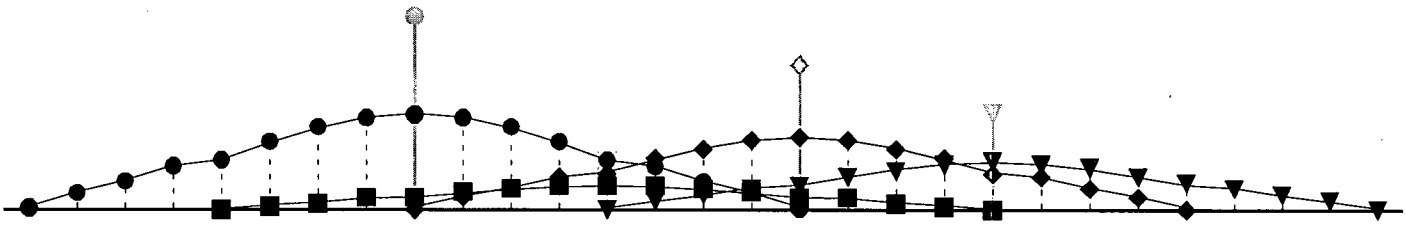
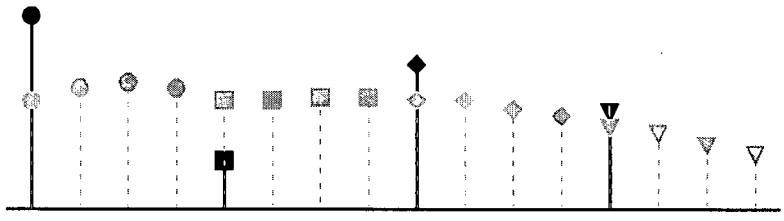


Figure 2.19 The original one dimension data

Mapping these data onto the expanded domain, then adding the overlapped values, the result of the expansion to the right direction is [**2.25** 2.5 2.8 2.56 **2.17** 2.28 2.35 2.27 **2.2** 2.12 2 1.9 **1.72** 1.6 1.5 1.3], **Figure 2.20** shows this process.



(a)



(b)

Figure 2.20 (a) The mapped data onto the expanded domain, (b) the expansion result to the right direction using Bézier interpolation

2.5.3 Two-Dimension Interpolation

Assume the original 2-dimentional image is to be expanded 4 times. In this case each pixel is mapped onto a 15x15 block then the overlapped values are added, then rounded to the nearest integer. The following is the result of the expansion to the right direction.

0	**20**	**40**	**20**
20	**60**	**80**	**40**
40	**80**	**60**	**20**
20	**40**	**20**	**80**

=>

7	11	14	18	**19**	22	24	26	**25**	25	23	20	**16**	15	11	8
11	17	21	26	27	30	33	34	32	33	29	26	20	18	14	10
16	21	26	31	32	37	39	40	38	32	34	29	23	21	16	11
18	26	31	37	38	43	45	46	42	38	38	33	25	23	17	12
19	27	32	38	**39**	44	46	46	**42**	42	38	32	**25**	23	17	12
22	31	37	43	44	50	51	52	47	47	43	37	28	26	20	14
24	33	39	45	46	51	53	53	48	48	43	37	29	27	21	15
26	34	40	46	46	52	53	52	47	48	43	38	29	27	21	16
25	32	38	42	**42**	47	48	47	**42**	43	39	34	**27**	25	19	14
25	33	38	42	42	47	48	48	43	44	40	36	29	27	21	16
23	29	34	38	38	43	43	43	39	40	38	34	28	26	21	16
20	26	29	33	32	37	37	38	34	36	34	34	26	25	20	15
16	20	23	25	**25**	28	29	29	**27**	29	28	26	**23**	21	18	14
15	18	21	23	23	26	27	27	25	27	26	25	21	20	17	13
11	14	16	17	17	20	21	21	19	21	21	20	18	17	14	10
8	10	11	12	12	14	15	16	14	16	16	15	14	13	10	9

2.5.4 Algorithm

Bézier interpolation algorithm is a separable process. That means it can be applied as a one-dimensional Bézier interpolation on every row and then as a one-dimensional Bézier interpolation on every column.

2.6 Spline Interpolation Method

The term spline goes back to the long flexible strips of metal used by draftsmen to lay out the surface of airplanes, cars, and ships. "Ducks", which are attached to the splines, were used to pull the spline in various directions. Splines are piecewise polynomials with pieces that are smoothly connected together. The connection points of the polynomials are called "knots". For a spline of degree n, each segment is a polynomial of degree n, which needs n + 1 coefficients to describe each piece [16, 60]. However, there is an additional smoothness constraint that imposes the continuity of the spline and its derivatives up to order (n – 1) at the knots [60]. Cubic splines are the most popular splines because they have minimum oscillation at the knots. Higher order splines produce more oscillations.

The cubic spline interpolation involves 4 points and it uses three different third-degree polynomials. This is different from the cubic interpolation. The cubic interpolation fits a single third-degree polynomial through the four points it uses to interpolate. Cubic spline fits three different polynomials, each is a third-degree polynomial, one between the first and the second points, one between the second and the third points, and one between the third and the fourth points. This method does not have a fixed model (kernel). The cubic spline polynomials are computed by solving a set of equations that involves the equations of the polynomials and their derivatives. This method is stable because it is data dependent, so that no fixed model (kernel) can be used for all points as in the other previous methods [16, 60]. The cubic spline polynomial produces relatively good edge

lines, however, it oscillates at the edge lines, but the major drawbacks of this method are the complexity and the computational time.

2.6.1 Mathematical Representation

Using four data points (pairs of numbers) (x_0, f_0), (x_1, f_1), (x_2, f_2), and (x_3, f_3) the cubic spline interpolation finds three cubic polynomials, $p_0(x)$ that interpolates between x_0 and x_1, $p_1(x)$ that interpolates between x_1 and x_2, $p_2(x)$ that interpolates between x_2 and x_3. For each polynomial there are four conditions and these are

$$p_j(x_j) = f_j, \quad p_j(x_{j+1}) = f_{j+1}, \quad p_j`(x_j) = k_j, \quad p_j`(x_{j+1}) = k_{j+1}, \qquad (14)$$

where $p_j`(x_j)$ is the derivative of $p_j(x_j)$, $j = 0, 1, 2$, and k_0 and k_3 are given, for simplicity they are equal to zero. By direct calculation the spline polynomial can be written as

$$p_j(x) = f_j * (x - x_{j+1})^2 [1 + 2(x - x_j)] + f_{j+1} * (x - x_j)^2 [1 - 2(x - x_{j+1})]$$
$$+ k_j * (x - x_j)(x - x_{j+1})^2 + k_{j+1} * (x - x_j)^2 (x - x_{j+1}) \qquad (15)$$

Differentiating $p_j(x)$ and $p_{j+1}(x)$ twice, and making

$$p_j``(x) = p_{j+1}``(x) \qquad (16)$$

results in

$$k_{j-1} + 4k_j + k_{j+1} = 3(f_{j+1} - f_{j-1}) \quad (17)$$

The spline polynomial could be shown to bealso

$$p_j(x) = a_{j0} + a_{j1}(x - x_j) + a_{j2}(x - x_j)^2 + a_{j3}(x - x_j)^3 \quad (18)$$

Using Taylor's formula, the spline coefficients (a_{j0}, a_{j1}, a_{j2}, and a_{j3}) are obtained as

$$\begin{aligned} a_{j0} &= p_j(x_j) = f_j \\ a_{j1} &= p_j'(x_j) = k_j \\ a_{j2} &= 3(f_{j+1} - f_j) - (k_{j+1} + 2k_j) \\ a_{j3} &= 2(f_j - f_{j+1}) - (k_{j+1} + k_j) \end{aligned} \quad (19)$$

The three cubic spline polynomials are written as [60]

$$\begin{aligned} p_0(x) &= a_{00} + a_{01}(x - x_0) + a_{02}(x - x_0)^2 + a_{03}(x - x_0)^3 \\ p_1(x) &= a_{10} + a_{11}(x - x_1) + a_{12}(x - x_1)^2 + a_{13}(x - x_1)^3 \\ p_2(x) &= a_{20} + a_{21}(x - x_2) + a_{22}(x - x_2)^2 + a_{23}(x - x_2)^3 \end{aligned} \quad (20)$$

where the a_j's (a_{j0}, a_{j1}, a_{j2}, and a_{j3}) are the spline coefficients.

The process starts with computing the derivatives of spline polynomials at the nodes of the spline curve (k_0, k_1, k_2, k_3). Usually k_0, and k_3 are given or set to zero. The k_1, and k_2 are found by solving the system of equations

$$k_0 + 4\,k_1 + k_2 = 3\,(f_2 - f_0)$$
$$k_1 + 4\,k_2 + k_3 = 3\,(f_3 - f_1) \qquad (21)$$

Next step is calculating the spline coefficients. Because this type of interpolation is data-dependent, so there is no fixed model for any arbitrary values.

2.6.2 One-dimension Interpolation

Let the original data set be [4 1 3 2], as shown in **Figure 2.21**, and assume it is to be expanded 4 times and $k_0 = k_3 = 0$.

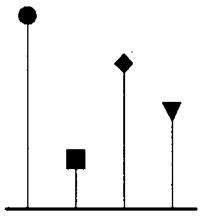


Figure 2.21 The original one dimension data

Solving the mathematical equations and interpolating, the result of the expansion to the right direction of the cubic spline interpolation method is [**4** 3.49 2.39 1.34 **1** 1.33 1.83 2.41 **3** 2.87 2.52 2.16 **2** 1.8 0.6 0.2], **Figure 2.22** shows the expanded result.

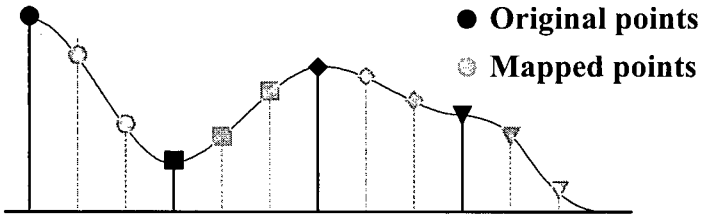


Figure 2.22 The result of 4-time expansion using spline interpolation

2.6.3 Two-Dimension Interpolation

Assume the original 2-dimentional image is to be expanded 4 times using bicubic spline interpolation. The following is the result of expansion to the right direction.

0	**20**	**40**	**20**
20	**60**	**80**	**40**
40	**80**	**60**	**20**
20	**40**	**20**	**80**

=>

0	3	8	15	**20**	26	33	38	**40**	36	25	23	**20**	18	8	3
3	5	12	19	25	31	38	44	46	41	34	27	23	20	10	4
8	12	20	29	37	43	51	57	58	53	43	34	31	27	13	5
15	19	29	41	50	57	65	71	72	65	53	42	37	32	14	5
20	25	37	50	**60**	67	75	80	**80**	71	58	46	**40**	36	16	6
26	31	43	57	67	74	79	82	80	71	56	42	36	31	14	5
33	38	51	65	75	79	81	79	75	64	48	34	28	25	12	4
38	44	57	71	80	82	79	74	67	56	41	27	21	19	9	3
40	46	58	72	**80**	80	75	67	**60**	50	37	25	**20**	16	7	2
36	41	53	65	71	71	64	56	50	45	39	34	32	20	10	4
25	34	43	53	58	56	48	41	37	39	44	49	52	36	14	6
23	27	34	42	46	42	34	27	25	34	49	65	71	59	20	8
20	23	31	37	**40**	36	28	21	**20**	32	52	71	**80**	68	24	10
18	20	27	32	36	31	25	19	16	20	36	59	68	29	13	6
8	10	13	14	16	14	12	9	7	10	14	20	24	13	7	3
3	4	5	5	6	5	4	3	2	4	6	8	10	6	3	1

2.6.4 Algorithm

The spline interpolation process is a separable one. That means it can be applied as a one-dimensional spline interpolation on every row and then a one-dimensional spline interpolation on every column.

2.7 B-Spline Interpola tion

B-spline is a special family of spline curves, the B is derived from the word basis. These splines consist of curve segments whose polynomial coefficients depend on the few middle points only. Thus, changing the value of a point affects only a small part of the curve, this behavior is called local control. In addition the time needed to compute the coefficients is greatly reduced. B-splines have the same continuity behavior as natural splines, but they do not pass through the original points. So there is a trade off between accuracy and simplicity. The natural splines produce relatively accurate curves but they are sensitive to the local values (pixels) and are computationally demanding. The B-spline method produces smoothed curves, which do not pass through the original points, but it is relatively a simple method [28, 63, 64]. The most famous B-spline interpolation method is the cubic B-spline interpolation method, which uses four points in the one dimensional case. A B-spline curve results from repeatedly convolving the Box function with itself. The order of the B-spline is the number of the times the box function is convolved with itself [28, 64]. The Cubic B-spline results from convolving the box function with itself three times. The cubic B-spline interpolation method is simpler and easier than the spline interpolation method, more stable than the cubic interpolation method because it does not oscillate, and computationally cheap. On the other hand, the original data are not exactly mapped, which means it is not as accurate. The other disadvantage is that the transition line at an edge is not as sharp as i.e. the cubic B-spline introduces smoothness to the expanded image [16].

2.7.1 Mathematical Representation

The cubic B-spline polynomial is the result of the convolution of a box function with itself three times. Given four equidistant points, the data (pairs of numbers) (x_{-1}, f_{-1}), (x_0, f_0), (x_1, f_1), and (x_2, f_2), assume $x_1 - x_0 = 1$, and $r = x - x_0$. The cubic B-spline function contribution expands to two neighbor pixels each side (forward and backward). The cubic spline contribution function can be give by the following equation

$$p(t) = \begin{cases} u(1-t) & (x_{-2} < t < x_{-1}) \\ v(2-t) & (x_{-1} < t < x_0) \\ v(t-2) & (x_0 < t < x_1) \\ u(t-3) & (x_1 < t < x_2) \end{cases}$$

where

$$u(t) = \frac{1}{6}(1-t)^3$$

$$v(t) = \frac{1}{6}(3t^3 - 6t^2 + 4)$$

The result of the cubic B-spline interpolation is that f_{-1} contributes $\{ ((1 - r)^3 / 6) f_{-1} \}$ to the other points to be interpolated, f_0 contributes $\{ (0.5 r^3 - r^2 + 2 / 3) f_0 \}$, f_1 contributes $\{ (0.5 (1 - r)^3 - (1 - r)^2 + 2 / 3) f_1 \}$, and f_2 contributes $\{ (r^3 / 6) f_2 \}$ [2, 3]. Then the expanded image will result by adding the overlapped mapped values [28, 64]. The contributions can be written as follows

$$p(r) = \frac{1}{6}\begin{Bmatrix} (1-r)^3 f_{-1} & + \\ (3r^3 - 6r^2 + 4) f_0 & + \\ (3(1-r)^3 - 6(1-r)^2 + 4) f_1 & + \\ r^3 f_2 & \end{Bmatrix}$$

In 4-time expansion case, the original value will be mapped onto fifteen values in the expanded result and this is called the mapping function, as shown in **Figure 2.23**.

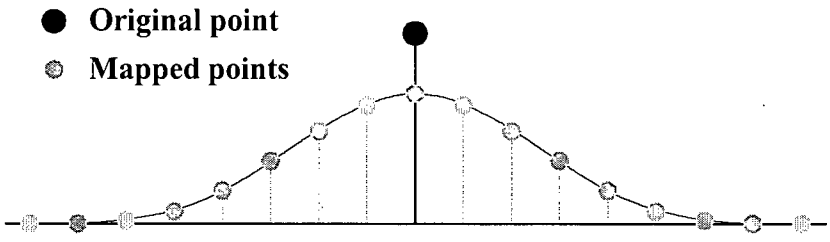


Figure 2.23 The original value, and its mapped result, in 4-time expansion case.

The two-dimensional mapping function can be found by simply convolving a horizontal one-dimensional mapping function with a vertical one-dimensional mapping function [63, 64]. As an example consider an expansion factor 4. A pixel in the original image is mapped onto a 15 × 15 block in the expanded image as shown in **Figure 2.24**.

$$1 \Rightarrow \frac{1}{64}\begin{bmatrix}
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & .1 & .1 & .1 & .1 & .1 & .1 & .1 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & .1 & .2 & .4 & .6 & .8 & 1 & .6 & .6 & .4 & .2 & .1 & 0 & 0 & 0 \\
0 & 0 & .1 & .3 & .8 & 1 & 2 & 3 & 3 & 3 & 2 & 1 & .8 & .3 & .1 & 0 & 0 \\
0 & 0 & .2 & .8 & 2 & 3 & 5 & 7 & 7 & 7 & 5 & 3 & 2 & .8 & .2 & 0 & 0 \\
0 & .1 & .4 & 1 & 3 & 6 & 10 & 12 & 13 & 12 & 10 & 6 & 3 & 1 & .4 & .1 & 0 \\
0 & .1 & .6 & 2 & 5 & 10 & 15 & 19 & 20 & 19 & 15 & 10 & 5 & 2 & .6 & .1 & 0 \\
0 & .1 & .8 & 3 & 7 & 12 & 19 & 24 & 26 & 24 & 19 & 12 & 7 & 3 & .8 & .1 & 0 \\
0 & .1 & 1 & 3 & 7 & 13 & 20 & 26 & \mathbf{28} & 26 & 20 & 13 & 7 & 3 & 1 & .1 & 0 \\
0 & .1 & .8 & 3 & 7 & 12 & 19 & 24 & 26 & 24 & 19 & 12 & 7 & 3 & .8 & .1 & 0 \\
0 & .1 & .6 & 2 & 5 & 10 & 15 & 19 & 20 & 19 & 15 & 10 & 5 & 2 & .6 & .1 & 0 \\
0 & .1 & .4 & 1 & 3 & 6 & 10 & 12 & 13 & 12 & 10 & 6 & 3 & 1 & .4 & .1 & 0 \\
0 & 0 & .2 & .8 & 2 & 3 & 5 & 7 & 7 & 7 & 5 & 3 & 2 & .8 & .2 & 0 & 0 \\
0 & 0 & .1 & .3 & .8 & 1 & 2 & 3 & 3 & 3 & 2 & 1 & .8 & .3 & .1 & 0 & 0 \\
0 & 0 & 0 & .1 & .2 & .4 & .6 & .8 & 1 & .8 & .6 & .4 & .2 & .1 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & .1 & .1 & .1 & .1 & .1 & .1 & .1 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0
\end{bmatrix}$$

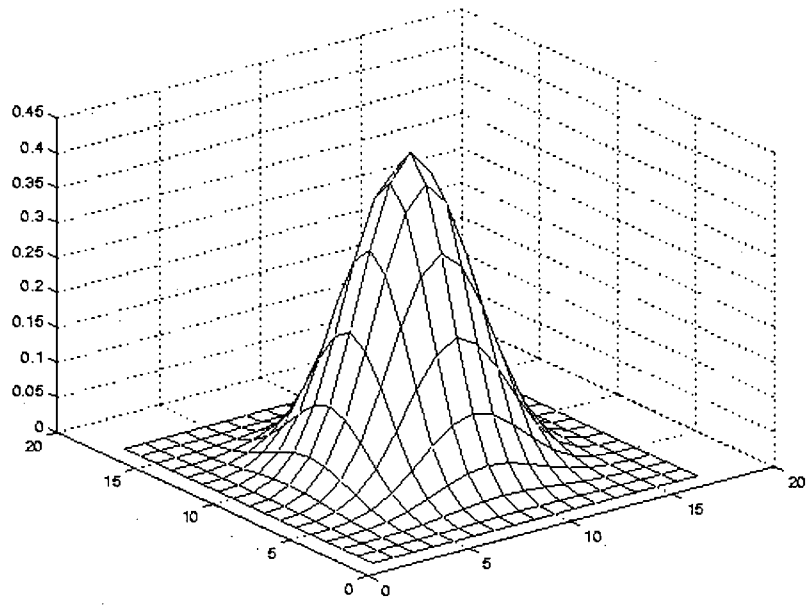


Figure 2.24 The model (spread function) of B-spline Interpolation

2.7.2 One-dimension Interpolation

Let the original data set be [4 1 3 2], as shown in **Figure 2.25**, and assume it is to be expanded 4 times.

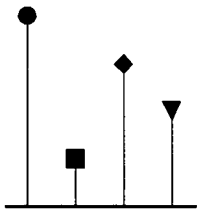
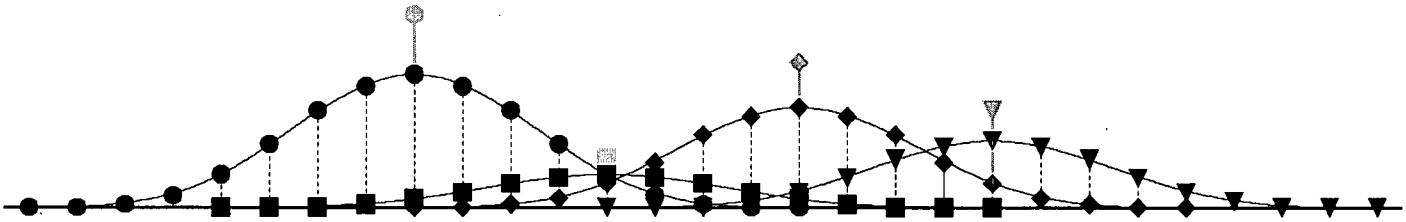
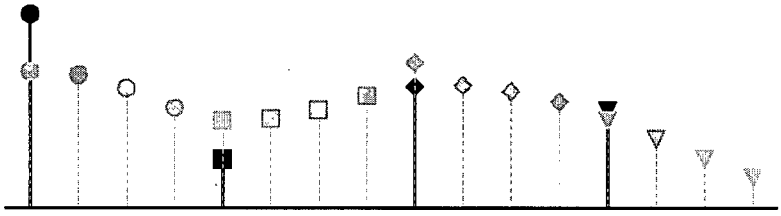


Figure 2.25 The original one dimension data

Mapping these data onto the expanded domain, then adding the overlapped values, the expanded result to the right direction is [**2.8** 2.77 2.46 2.08 **1.83** 1.84 2.04 2.3 **2.5** 2.54 2.42 2.17 **1.83** 1.44 1.02 0.64], **Figure 2.26** shows this process.



(a)



(b)

Figure 2.26 (a) The mapped data onto the expanded domain, (b) the result of adding the overlapped values

2.7.3 Two-Dimension Interpolation

Assume the original 2-dimentional image is to be expanded 4 times. In this case each pixel is mapped onto a 15x15 block then the overlapped values are added, then rounded to the nearest integer. The following is the result of the expansion in the right direction.

0	**20**	**40**	**20**
20	**60**	**80**	**40**
40	**80**	**60**	**20**
20	**40**	**20**	**80**

=>

6	10	14	18	**23**	27	31	33	**34**	32	29	25	**20**	15	10	6
10	14	19	25	30	35	40	42	43	41	36	31	25	19	13	8
14	19	25	32	38	44	48	51	51	48	43	36	29	22	15	9
18	25	32	39	46	52	56	58	57	54	48	40	32	24	17	10
23	30	38	46	**53**	58	62	63	**62**	57	51	43	**34**	25	17	11
27	35	44	52	58	63	65	66	63	58	51	42	33	25	17	10
31	40	48	56	62	65	67	66	62	56	49	41	32	24	16	10
33	41	51	58	63	66	66	63	59	53	47	39	31	23	16	10
34	43	51	57	**62**	63	62	59	**55**	50	45	38	**32**	24	17	11
32	41	48	54	57	58	56	52	50	47	44	40	35	28	20	13
29	36	43	48	51	51	49	47	45	44	44	43	39	33	24	15
25	31	36	41	43	42	41	39	38	40	43	44	42	36	27	18
20	25	29	32	**34**	33	32	31	**32**	35	39	42	**42**	36	28	18
15	19	22	24	25	25	24	23	24	28	33	36	36	32	24	16
10	13	15	17	17	17	16	16	17	20	24	27	28	24	19	12
6	8	9	10	11	10	10	10	11	13	15	18	18	16	12	8

2.7.4 Algorithm

The cubic B-spline interpolation process is a separable one. That means it can be applied as a one-dimensional cubic B-spline interpolation on every row and then as a one-dimensional cubic B-spline interpolation on every column.

Frequency Interpolation

This method starts with expanding the signal using the replication method. Then a low pass filter is applied in the frequency domain to smooth the high frequencies (blockiness effect) that are introduced by the replication expansion. This method is also called defocusing [41]. It sounds easy but there is one thing to be careful with, and that is the shape of the filter. There are many low pass filters that could be applied in this type of interpolation, but only two cases are discussed in this section: (1) The ideal low pass filter; and (2) The Butterworth low pass filter.

2.8 Ideal Filter Interpolation

After the pixel replication method is used to expand the image, an ideal low pass filter is applied to the FFT of that expanded image [23]. The cutoff frequency is f_{max}, where f_{max} is the maximum frequency of the original image. It is clear that this method is simple and very fast. However, this method causes blurring, rippling and oscillation effects at the edges in the expanded image [27, 41].

2.8.1 One-dimension Interpolation

Let the original data set be [4 1 3 2], as shown in **Figure 2.27**, and assume it is to be expanded 4 times. The expanded points using replication are [**4** 4 4 4 **1** 1 1 1 **3** 3 3 3 **2** 2 2 2], as shown in **Figure 2.28**.

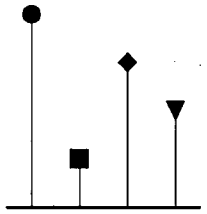


Figure 2.27 The original one dimension data

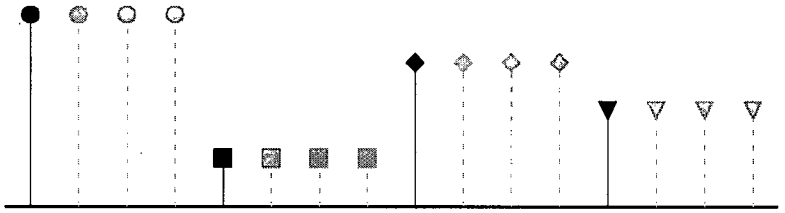


Figure 2.28 The expanded one dimension data using replication method

Applying an ideal low pass filter on the FFT of the expanded data is shown in **Figure 2.29**. Then computing the IFFT of the filtered points, the expanded result is [**3.44** 4.28 4.28 3.25 **1.75** 0.72 0.72 1.56 **2.56** 3.13 3.13 2.75 **2.25** 1.87 1.87 2.44], where the original data are bolded for convenience. **Figure 2.30** shows this process.

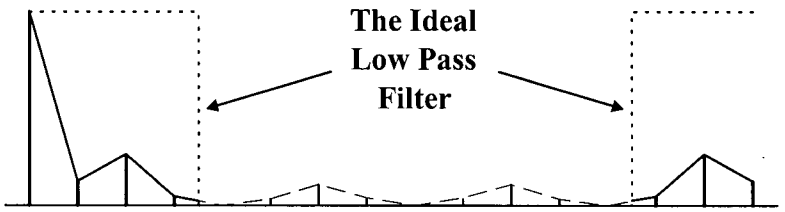


Figure 2.29 The FFT of the expanded signal, with the Ideal LPF, and filtered FFT.

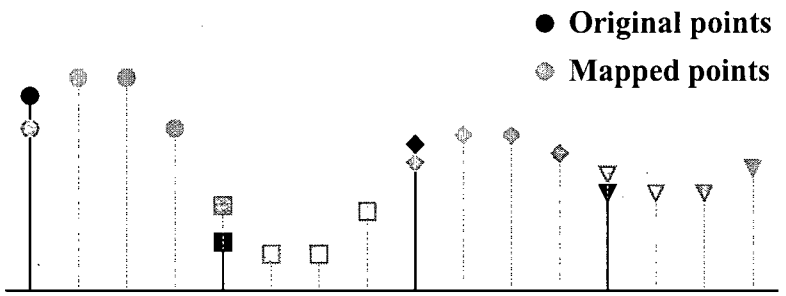


Figure 2.30 The result of IFFT of the filtered signal by using ideal LPF

2.8.2 Two-Dimension Interpolation

Assume the original 2-dimentional image is to be expanded 4 times, the expanded result as follows:

0	**20**	**40**	**20**
20	**60**	**80**	**40**
40	**80**	**60**	**20**
20	**40**	**20**	**80**

=>

14	3	3	13	**23**	28	28	29	**34**	39	39	35	**34**	35	35	28
3	0	0	3	12	15	15	20	31	42	42	32	20	16	16	12
3	0	0	3	12	15	15	20	31	42	42	32	20	16	16	12
13	3	3	14	26	32	32	34	43	52	52	42	31	26	26	23
23	12	12	26	**44**	52	52	54	**62**	73	73	60	**43**	35	35	33
28	15	15	32	52	63	63	65	76	90	90	72	51	39	39	38
28	15	15	32	52	63	63	65	76	90	90	72	51	39	39	38
29	20	20	34	54	65	65	63	68	76	76	63	45	34	34	35
34	31	31	43	**62**	76	76	68	**63**	65	65	54	**36**	23	23	30
39	42	42	52	73	90	90	76	65	63	63	52	29	10	10	25
39	42	42	52	73	90	90	76	65	65	65	52	29	10	10	25
35	32	32	42	60	72	72	63	54	52	52	47	39	32	32	35
34	20	20	31	**43**	51	51	45	**36**	29	29	39	**55**	68	68	54
35	16	16	26	35	39	39	34	23	10	10	32	68	96	96	69
35	16	16	26	35	39	39	34	23	10	10	32	68	96	96	69
28	12	12	23	33	38	38	35	30	25	25	35	54	69	69	51

2.8.3 Algorithm

This method is not a separable method.

2.9 Butterworth Filter Interpolation

A Butterworth low pass filter is used instead of an ideal low pass filter. The filter is applied on the FFT of the data expanded using the replication method. This method produces better results than the ideal low pass filter, in terms of ripples and oscillations [23, 27]. But it is not as simple to implement.

2.9.1 One-dimension Interpolation

Let the original data set be [4 1 3 2], as shown in **Figure 2.31**, and assume it is to be expanded 4 times.

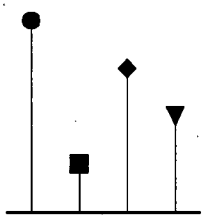


Figure 2.31 The original one dimension data

The expanded points using replication are [**4** 4 4 4 **1** 1 1 1 **3** 3 3 3 **2** 2 2 2], the original data is bolded for convenience, as shown in **Figure 2.32**.

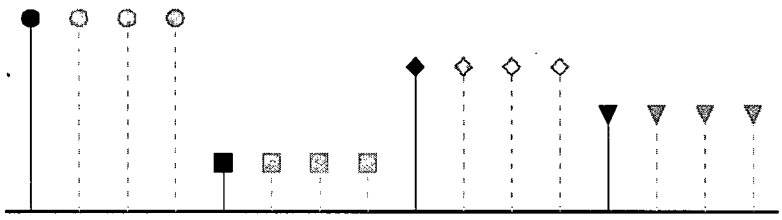


Figure 2.32 The expanded one dimension data using replication method

Applying the Butterworth low pass filter on the FFT of the expanded data as shown in **Figure 2.33**. Then computing the IFFT of the filtered points, the expanded result is [**3.48** 4 4 3.25 **1.75** 1 1 1.52 **2.52** 3 3 2.75 **2.25** 2 2 2.48], the original data is bolded for convenience. **Figure 2.34** shows this process.

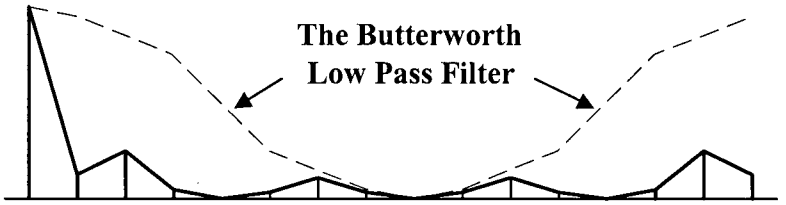


Figure 2.33 The FFT of the expanded signal, the Butterworth LPF

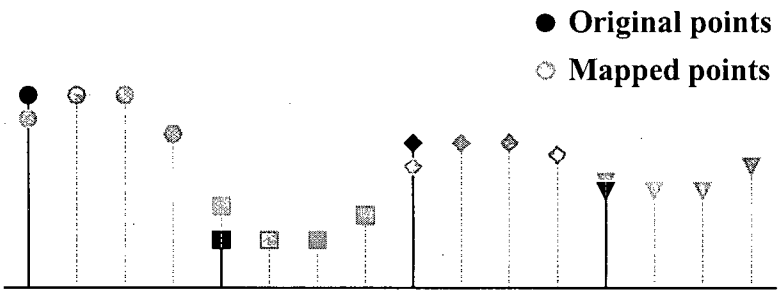


Figure 2.34 The result of IFFT of the signal filtered by the Butterworth LPF

2.9.2 Two-Dimension Interpolation

Assume the original 2-dimentional image is to be expanded 4 times. The expanded result after replication and applying the Butterworth filter is

0	**20**	**40**	**20**
20	**60**	**80**	**40**
40	**80**	**60**	**20**
20	**40**	**20**	**80**

=>

13	5	5	11	**21**	26	26	28	**33**	36	36	35	**34**	35	35	28
5	0	0	5	15	20	20	24	34	40	40	34	25	20	20	45
5	0	0	5	15	20	20	24	34	40	40	34	25	20	20	45
11	5	5	12	25	31	31	35	44	50	50	43	31	25	25	21
21	15	15	25	**42**	51	51	54	**64**	70	70	61	**44**	35	35	31
26	20	20	31	50	60	60	64	74	81	81	70	49	40	40	36
26	20	20	31	50	60	60	64	74	81	81	70	49	40	40	36
28	24	24	35	54	64	64	66	71	74	74	64	45	35	35	34
33	34	34	44	**64**	74	74	71	**66**	64	64	55	**35**	25	25	29
36	40	40	50	70	81	81	74	64	60	60	51	31	20	20	26
35	40	40	50	70	81	81	74	64	60	60	51	31	20	20	26
34	34	34	43	61	70	70	64	55	51	51	47	39	34	34	35
34	25	25	31	**44**	49	49	45	**35**	31	31	39	**56**	65	65	54
35	20	20	25	35	40	40	35	25	20	20	34	65	80	80	65
35	20	20	25	35	40	40	35	25	20	20	34	65	80	80	65
28	15	15	21	31	36	36	34	29	26	26	35	54	65	65	52

2.9.3 Algorithm

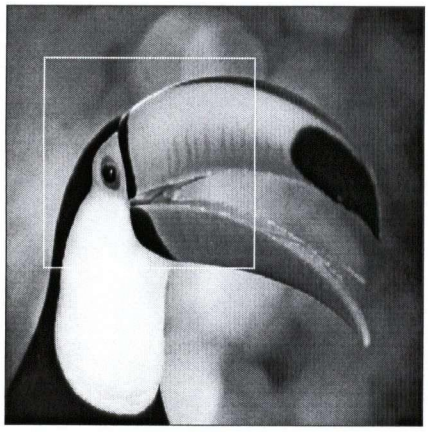
This method is not a separable method

3 Experimental Simulation

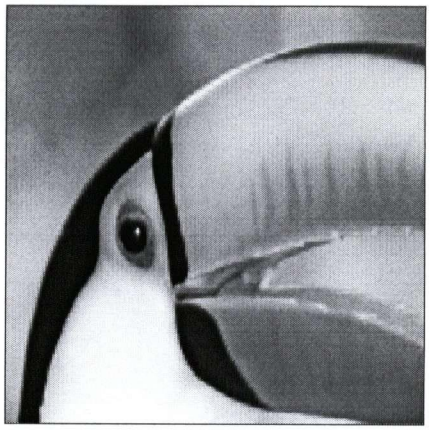
3.0 Introduction

In this section the results of using ten different methods, nine of which were discussed in the previous section. The tenth method is our proposed method and this is introduced and discussed in Chapter 4. For all these pictures, the expansion factor is equal to 2. The aim of this study is not to restore undersampled (scaled down) images, but to expand any given image. Thus the criterion for the effectiveness of each method is to be measured by the visual appearance as judged by the human observer.

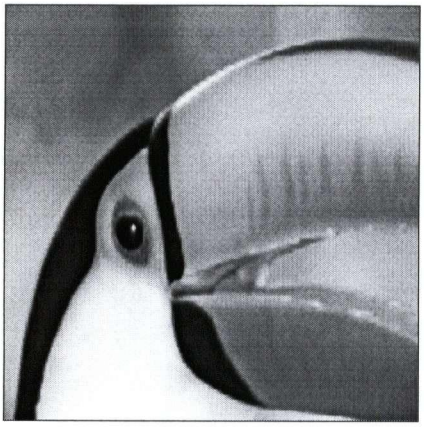
3.1 The Tropic Image Examples



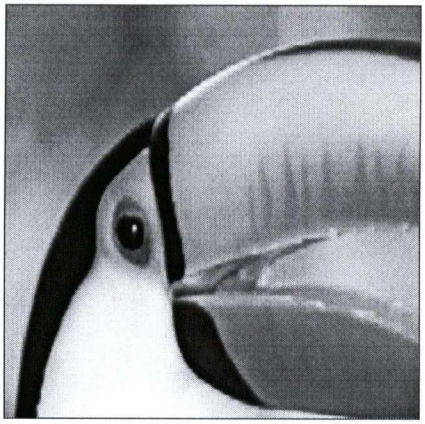
Original Image



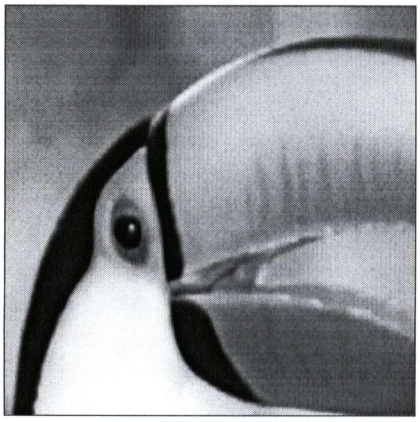
Replication



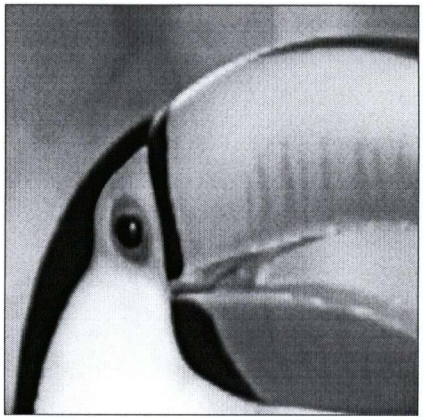
Linear Interpolation



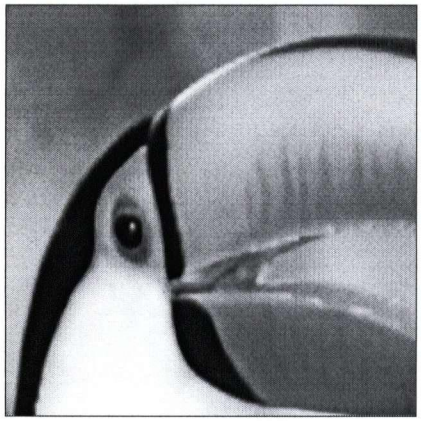
Quadratic Interpolation



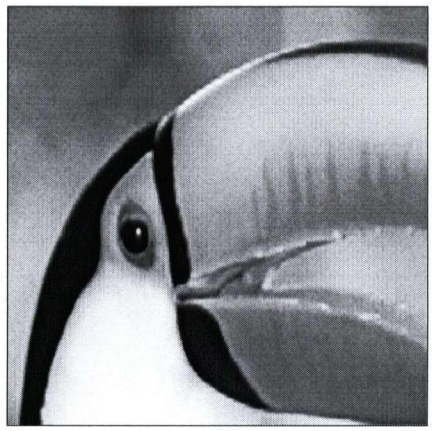
Cubic Interpolation



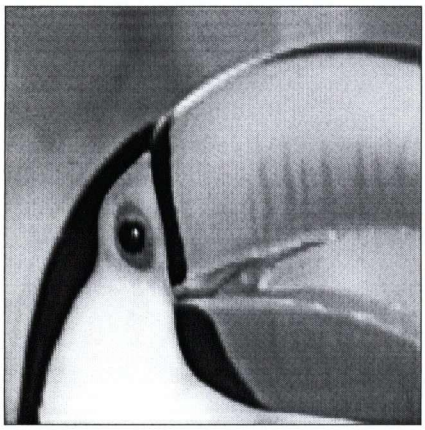
Bézier Interpolation



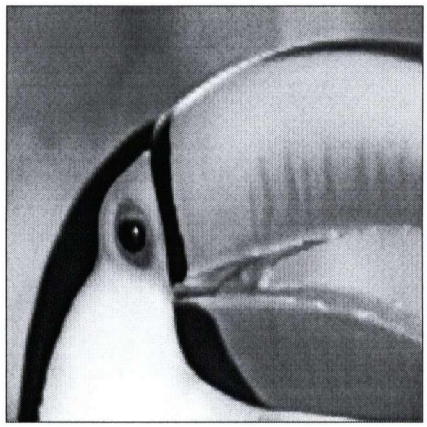
B-spline Interpolation



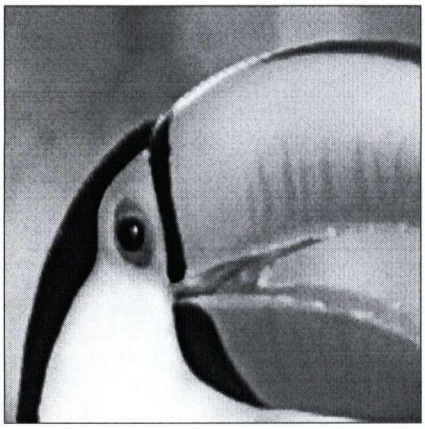
Spline Interpolation



Ideal LPF Frequency Interpolation



Butterworth LPF Frequency Interpolation



The Proposed Method

Figure 3.1 The "Tropic" image and its expanded results using ten different methods including the proposed method.

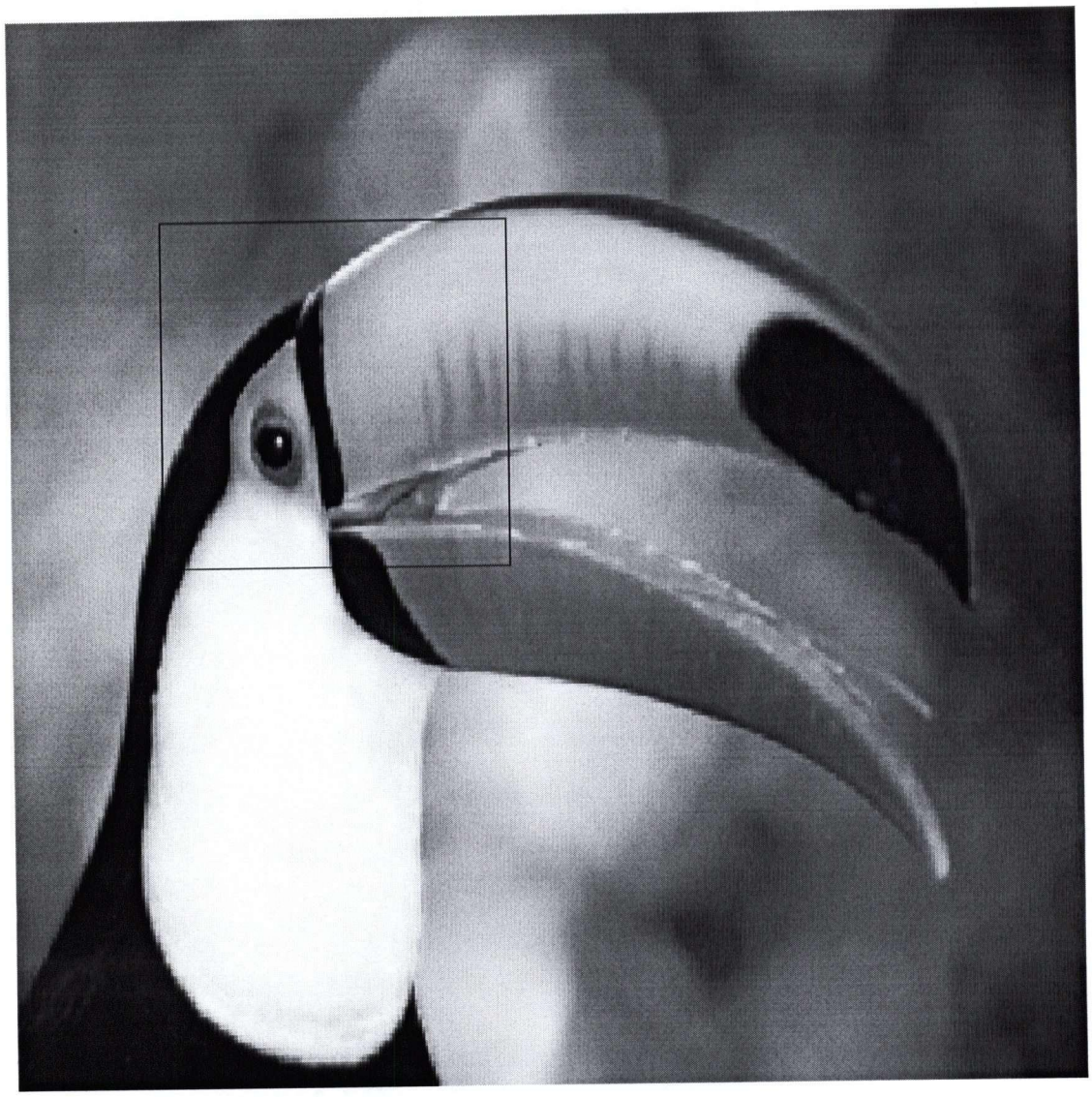


Figure 3.2 The original "Tropic" image

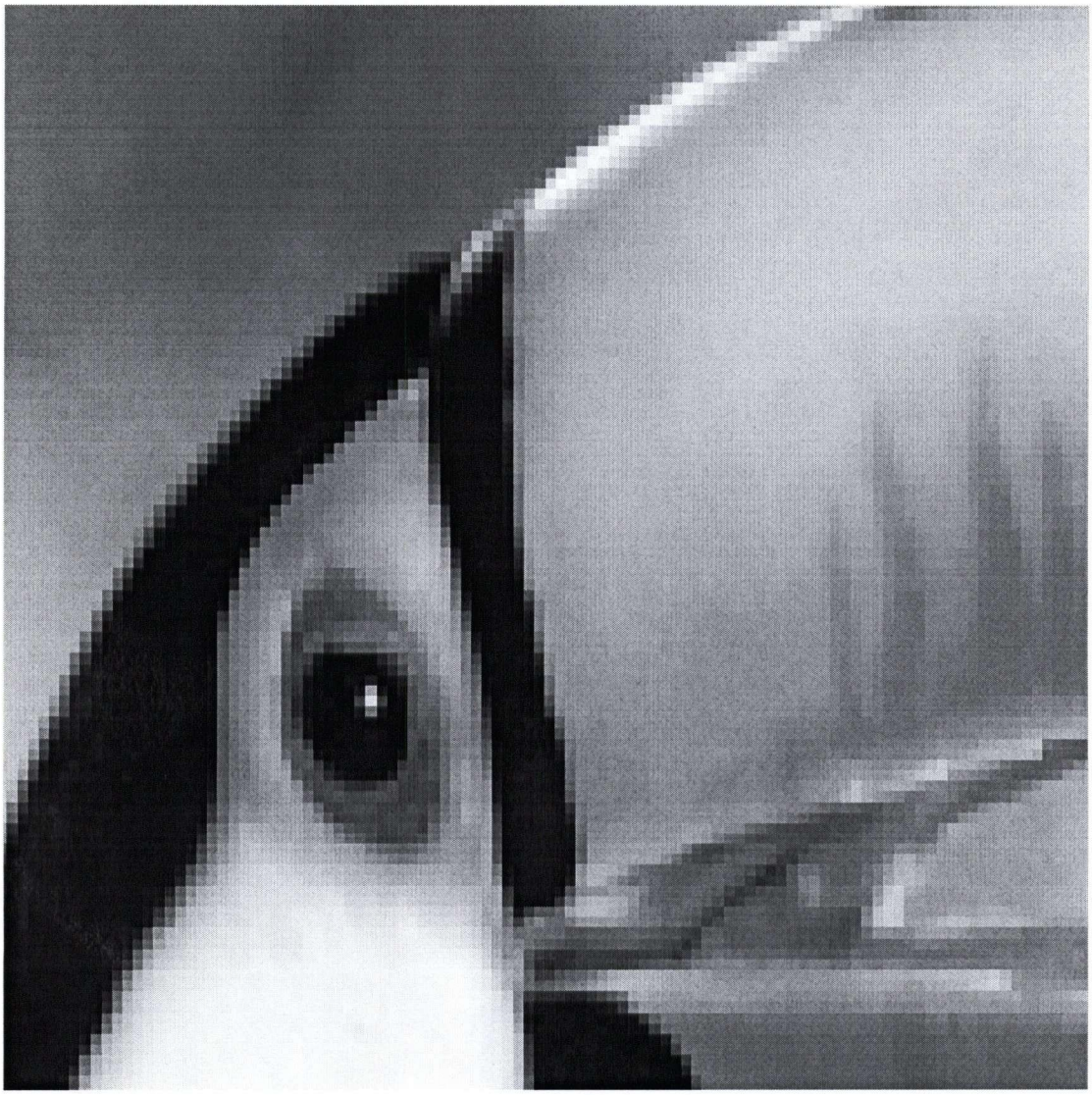


Figure 3.3 The Expanded image by the replication method, expansion factor is equal 3.

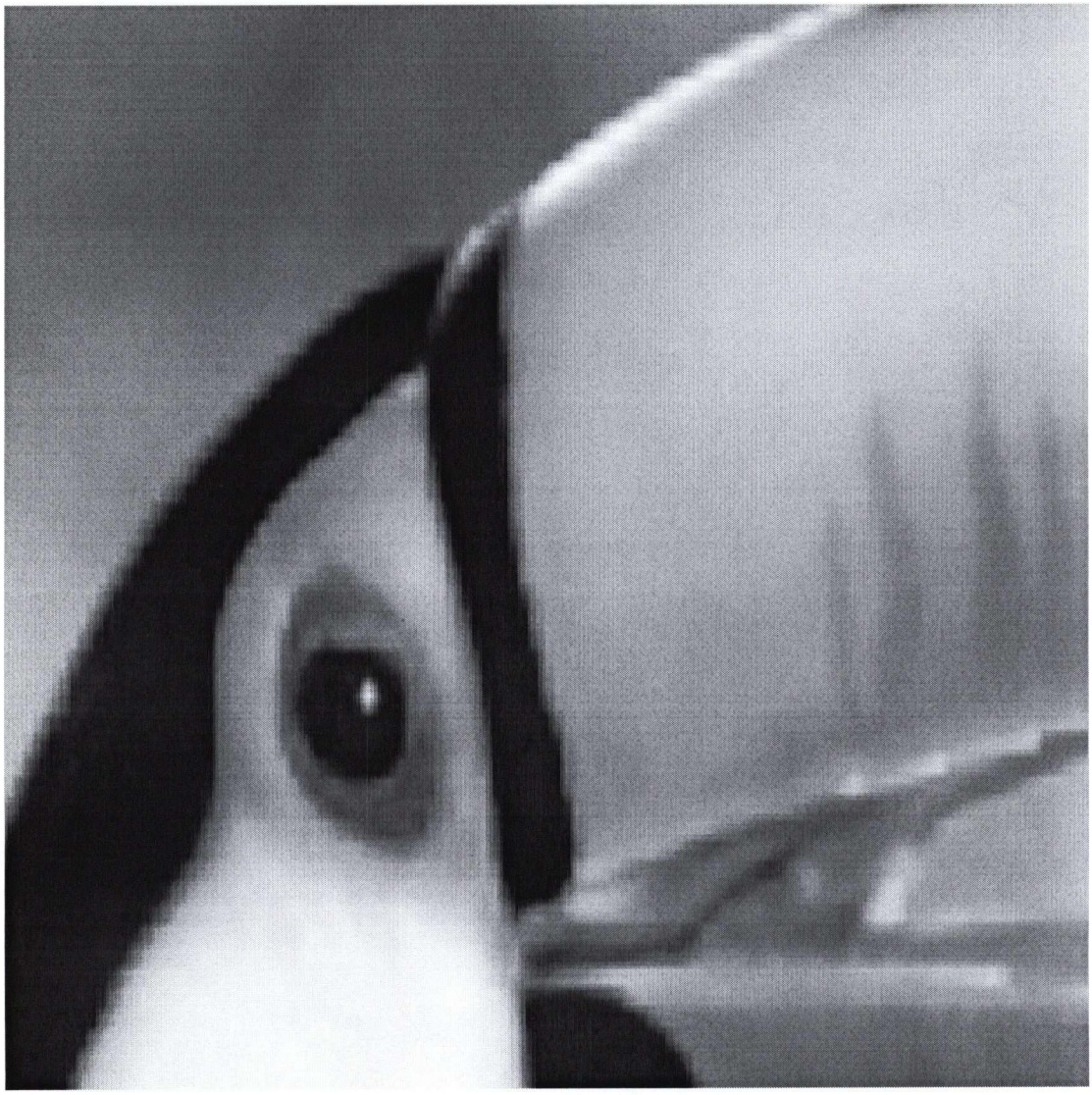


Figure 3.4 The Expanded image by the Linear Interpolation method, expansion factor is equal 3.

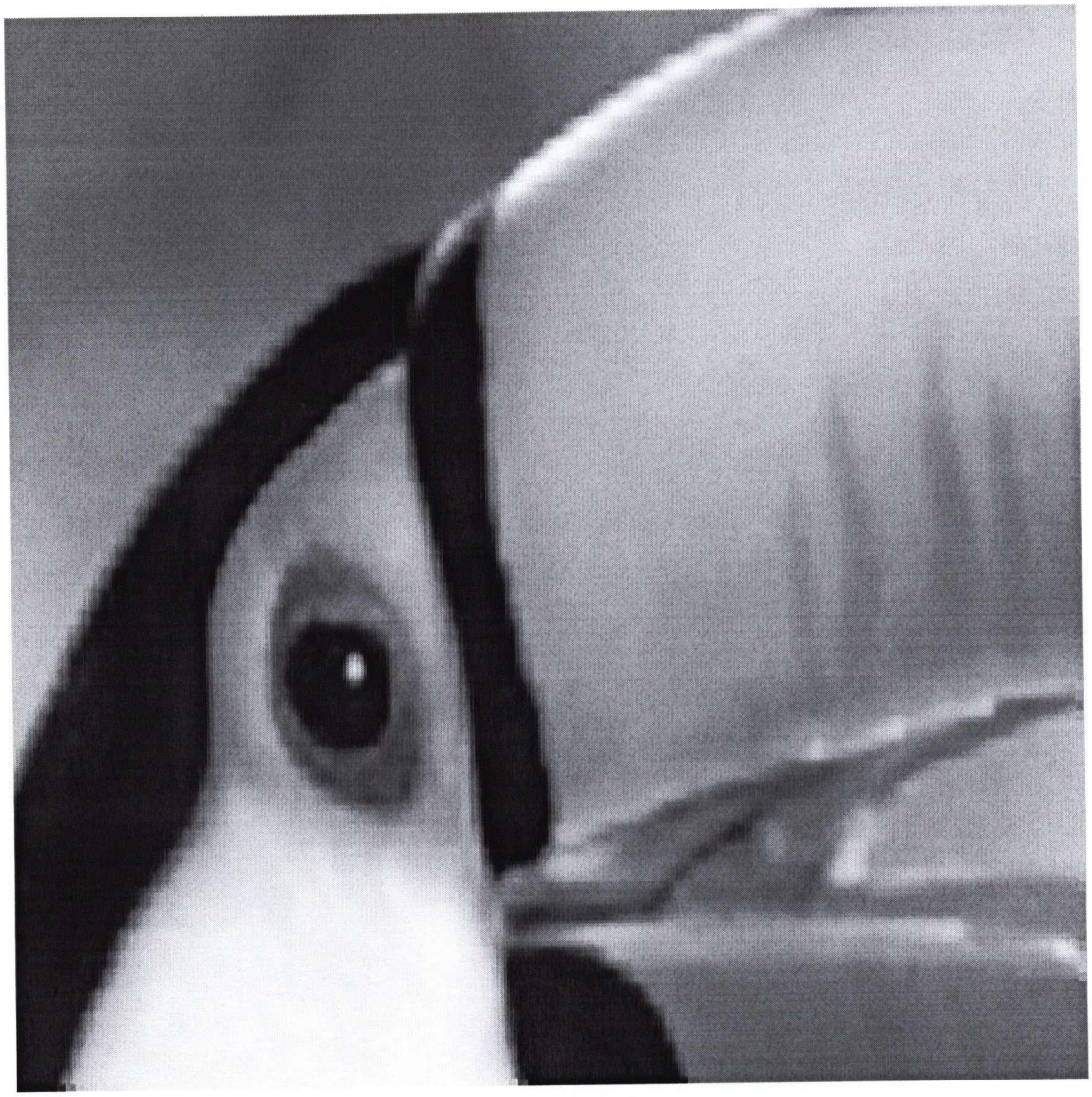


Figure 3.5 The Expanded image by the Cubic Interpolation method, expansion factor is equal 3.

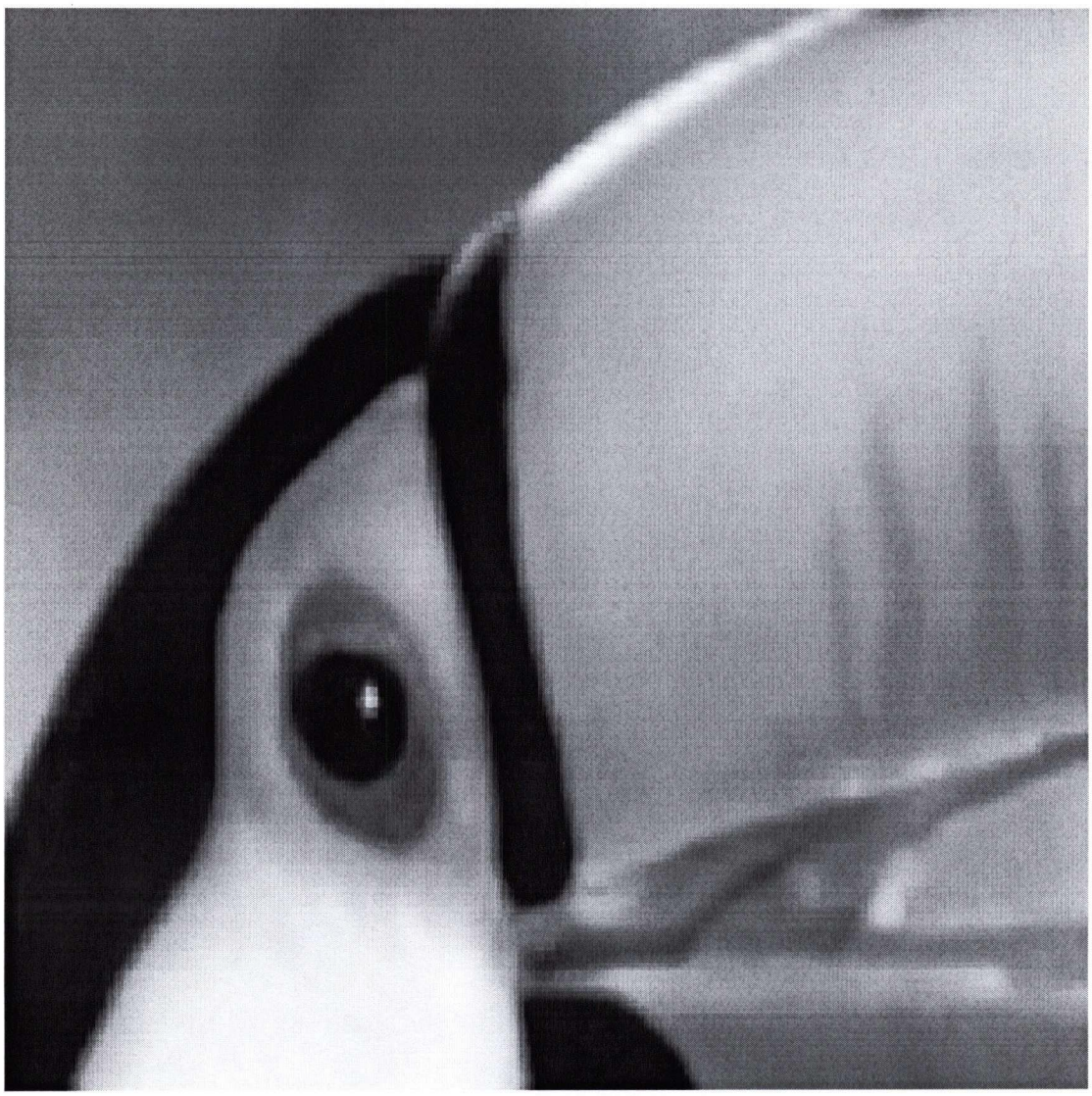
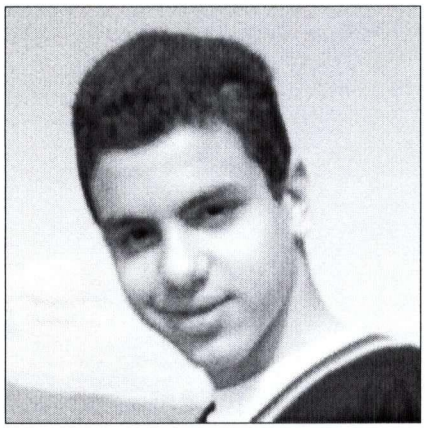


Figure 3.6 The Expanded image by the Proposed method, expansion factor is equal 3.

3.2 The Saad Image Examples

Original Image

Replication



Linear Interpolation

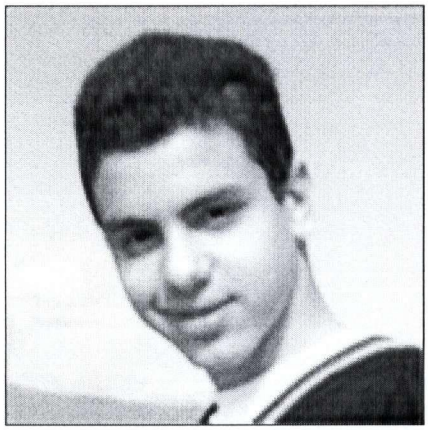
Quadratic Interpolation

Cubic Interpolation

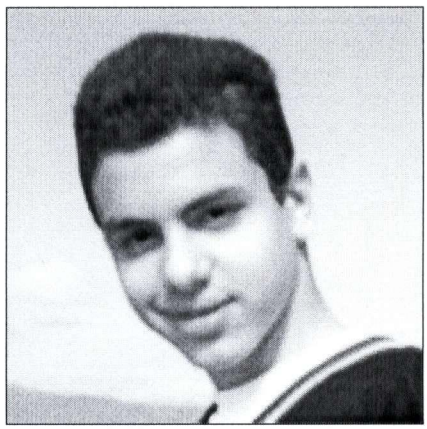
Bézier Interpolation

B-spline Interpolation

Spline Interpolation



Ideal LPF Frequency Interpolation



Butterworth LPF Frequency Interpolation

The Proposed Method

Figure 3.7 The "Saad" image and

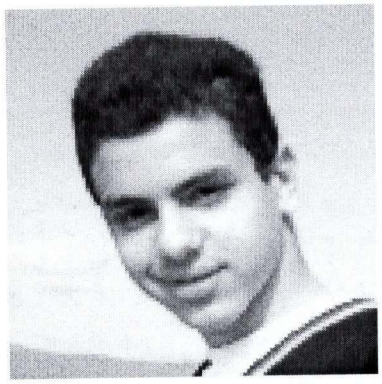


Figure 3.8 The original "Saad" image

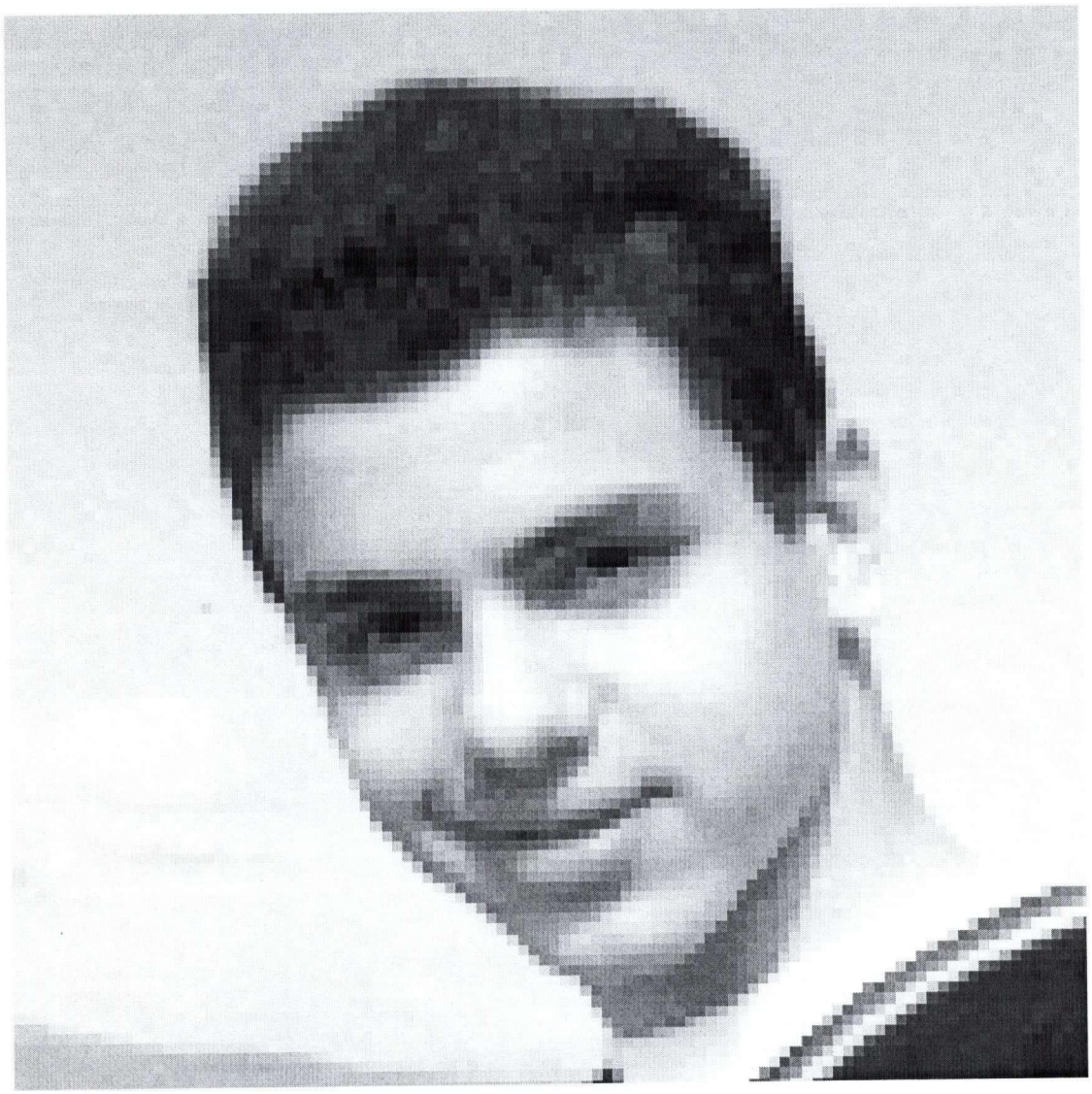


Figure 3.9 The Expanded image by the replication method, expansion factor is equal 3.

Figure 3.10 The Expanded image by the Linear Interpolation method, expansion factor is equal 3.

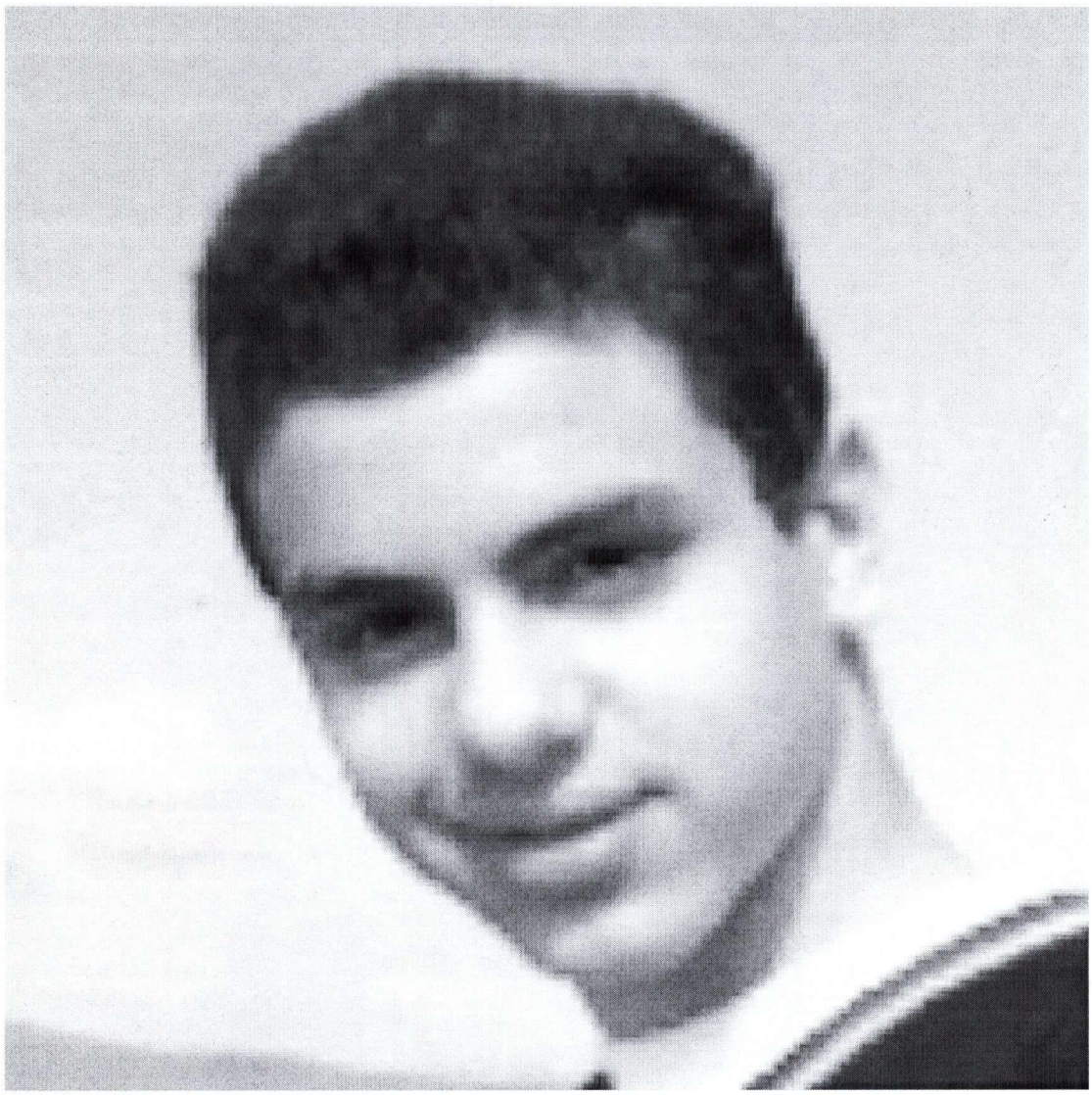


Figure 3.11 The Expanded image by the Cubic Interpolation method, expansion factor is equal 3

Figure 3.12 The Expanded image by the Proposed method, expansion factor is equal 3.

3.3 Analysis of the Expansion Methods

3.3.1 Replication Method

The pixel replication method is very fast and computationally cheap. Having these advantages makes it the first choice in many digital signal processing applications. The major drawback of the pixel replication method is the apparent blockiness effects introduced all over the image and specially at the edges [27, 41]. The blockiness effect is obvious at the edge whether the edge is sharp or smooth. A sharp edge is an edge where the intensity difference across it is large, and usually lies between two different regions in the image. A smooth edge is an edge where the intensity difference across it is small, and lies within a region in the image. The higher the expansion factor the more obvious and annoying the blockiness effects in the expanded image [1, 2, 3, 15]. Sometimes the blockiness effects are not acceptable even within a smooth region.

3.3.2 Linear Interpolation Method

The linear interpolation method assumes that the image has a continuos intensity function but not necessarily a continuos first derivative. The linear interpolation method is fast and computationally cheap, which makes it attractive for many applications and a good competitor to the pixel replication method in many image processing applications [4, 5, 41]. The linear interpolation method reduces but does not eliminate the blockiness effects at the edges introduced by the expansion process. However, it reduces the blockiness effects in the other areas of the expanded image. The blockiness and the blurring effects are obvious in the expanded image and the higher the expansion factor

the more obvious and annoying the blockiness effects. The linear interpolation method introduces a ringing or contouring effects as it smoothes the expanded image [27].

3.3.3 Quadratic Interpolation Method

The quadratic interpolation method assumes that the image has a continuos intensity function and a continuos first derivative but not necessarily a continuos second derivative [60]. This method reduces the ringing or contouring effects of the linear interpolation but it does not eliminate the blockiness effects. The quadratic interpolation method produces blurry expanded image but better than linear interpolation. It is more computationally demanding than the linear interpolation method. For higher expansion factors this method gives better performance than linear interpolation.

3.3.4 Cubic Interpolation Method

The cubic interpolation method assumes that the image has a continuos intensity function and continuos first and second derivatives but not necessarily continuos higher derivatives. The cubic interpolation method reduces the blockiness effects specially within homogeneous regions [60]. However, the blockiness at the edges still exists. Cubic interpolation produces better results than the linear and quadratic interpolation methods, but it is not as fast computationally. Cubic interpolation may introduce oscillation effects at the edges because of the assumption about the continuity of the image and its derivatives [15, 16].

3.3.5 Bézier Interpolation Method

Bézier interpolation eliminates any blockiness effects in the expanded image by (unacceptably) oversmoothing the expanded image. This method does not map the original pixels' values into the expanded image, so that it is not suitable for image expansion application. Bézier interpolation method is a good method for modeling and rendering in the computer vision fields [63, 64].

3.3.6 Spline Interpolation Method

Spline interpolation is sensitive to the pixels' values, because it is data dependent. This method interpolates the homogenous regions well [16]. On the other hand, it causes oscillation effects at the edges in the expanded image, which is not tolerable in some applications. The main drawback of spline interpolation is that it is very computationally expensive because it has to solve a set of spline equations for 4 pixel. In some applications, where the system complexity and cost of computations are not the first priorities, spline interpolation method could be considered a good interpolation method [15, 16].

3.3.7 B-Spline Interpolation Method

The Cubic B-spline method eliminates the blockiness in the expanded image by smoothing the expanded image, while it does not map the original pixels' values into the expanded image [63, 64]. Cubic B-spline smoothness is much less than that of the cubic Bézier method. The smoothness causes the sharp edges to seem defocused or blurred. This method deals very well with smooth areas inside the homogeneous regions. In

many applications this method is considered as a good method because it doesn't introduce blockiness effects as most other methods do [16].

3.3.8 Ideal Filter Interpolation Method

The ideal low pass filter cutoff frequency is equal to the original image bandwidth. That means the expanded image would only contain frequency components that exist in the original image [27]. Using this ideal low pass filter eliminates all the high frequency components from the expanded image. This introduces little distortion to the edges and ripples or oscillations (waves) effects in the homogenous regions. The sharp edges are relatively reserved, but the oscillation blurs the smooth areas inside the homogeneous regions [23]. The smoother the region the higher are the oscillation effects. As a result, the smoothest regions suffer the most. This method is computationally cheap and fast but neither precise nor efficient. This is not used for image expansion purposes.

3.3.9 Butterworth Filter Interpolation Method

Using other ideal low pass filters instead of the ideal low pass filter will better preserve the high frequency components. This will improve the ripple and oscillation effects, but introduces blockiness effects. The closer the low pass filter to the ideal low pass filter the more the oscillation and the less the blockiness, on the other hand the wider the transition period of the low pass filter the less the oscillations and the more the blockiness [23]. The extreme case is keeping all frequency components which result in the replication method.

4 Information-Preserving Image Expansion Method

4.0 Introduction

A common problem shared by existing methods is that the edges in the expanded image have a zigzag effect. This is specially unacceptable when these edges are known to be straight edges. Our method described in this chapter does not have this problem. After finding the edges in the original image, these edges are expanded so as to preserve their original shapes. Then the values of the pixels and around the edges are estimated so as to yield a natural and usually acceptable effect. This method starts with edge detection. There are many edge detection techniques that can be used to produce the edge image. These include the Sobel, Roberts, Laplacian [65], Canny [66, 67], Gaussian, or LoG (Laplacian of Gaussian) [68, 69, 70] edge detectors. The Laplacian edge detector is used with binary images because it produces closed edges in the edge image [66]. And Canny edge detector is used with gray level images because it produces a thin edge, which is of

single-pixel width [67]. There are many other methods that detect edge in binary images [71], gray-level images [72, 73, 74, 75], and color images [76, 77]. Other methods produce two or more pixel wide edges and this may cause problems in the expansion stage.

4.1 Edge Detection, Edge Encoding

1. Edge Detection

For gray-level images we use the Canny edge detector. For binary images, the 4-connectivity Laplacian operator [66], as shown in **Figure 4.1**, is used as an edge detection algorithm. This edge detector involves applying the Laplacian operator on a binary image. A positive, zero, or negative value will be the result of this edge detector. Only the positive outcomes are considered as an edge.

0	**-1**	**0**
-1	**4**	**-1**
0	**-1**	**0**

Figure 4.1 The Laplacian operator.

- The positive value occurs when the detected pixel has the value 1, and one or more of its 4-connectivety neighbor pixels have the value 0.
- The zero value occurs when the detected pixel and all of its 4-connectivety neighbor pixels have the same intensity values, either the value 1 or 0.
- The negative value occurs when the detected pixel has the value 0 and one or more of its 4-connectivety neighbor pixels have the value 1.

An example on the three possible values of the Laplacian edge detector that applied on a binary image is shown **Figure 4.2**.

2. Chain coding

After detecting the one pixel wide edges, these edges are coded using chain coding. All edges in the edge image are coded using an 8-connectivety chain code with a simple numbering scheme [78]. **Figure 4.3** shows the used numbering scheme.

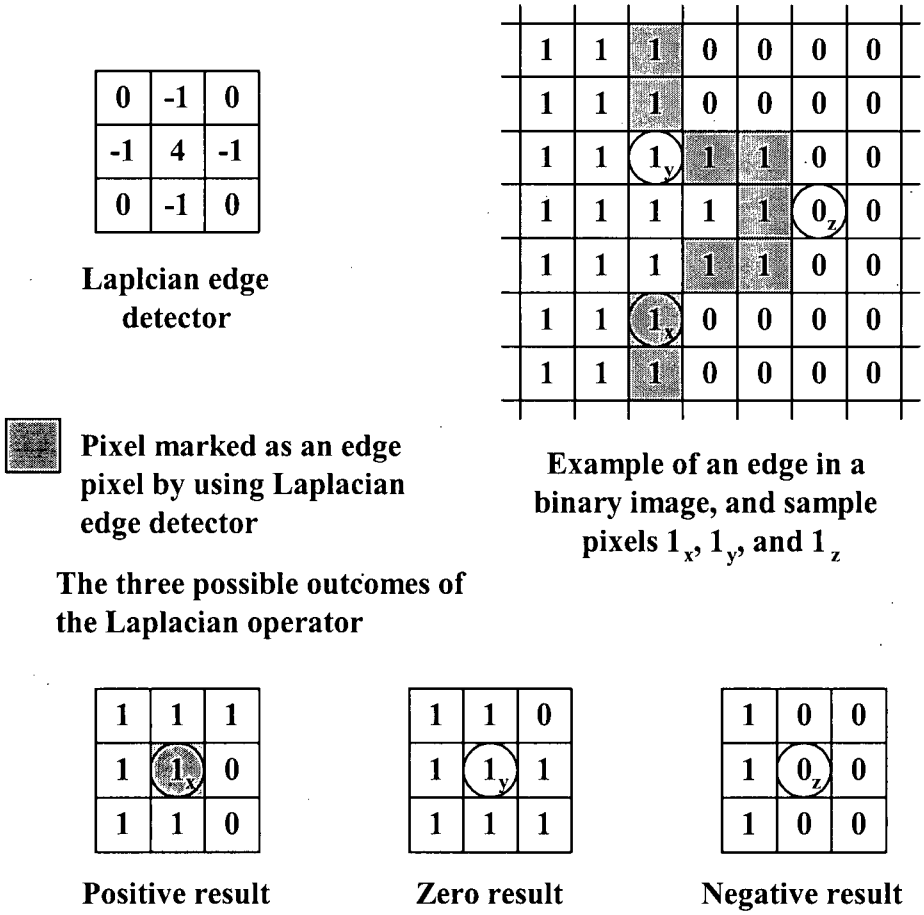


Figure 4.2 Example on an edge detected using the Laplacian edge detector.

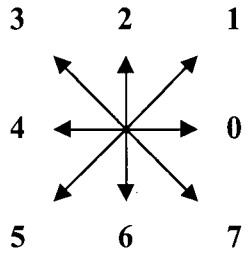


Figure 4.3 The numbering scheme used with 8-connectivety chain coding.

Chain coding starts from the upper-left corner to the lower-right corner. Chain coding of edges proceeds as follows:

1. The search for the beginning of an edge starts from right, left, up, down, north-east, north-west, south-west, and south-east, which is in chain code numbers: 0, 4, 2, 6, 1, 3, 5, then 7.
2. At each pixel position lying on an edge, the encoder starts the search for the next pixel using the previous pixel direction. For example, if the previous pixel on the edge is coded in the direction 6, which is the downward direction, the encoder then searches for the next chain pixel by considering the pixels whose direction are $6 + 1 = 7$ and $6 - 1 = 5$. If the pixel is an edge pixel i.e., with a pixel value in the edge image equals to 1, then the encoder stops the search for the next pixel.
3. If the pixel in step 2 is not an edge pixel, the encoder searches the remaining unchecked chain code directions in the following order: 0, 4, 2, 6, 1, 3, 5, then 7. In the above example the encoder will check the following sequence: 0, 4, 2, 1, then 3.

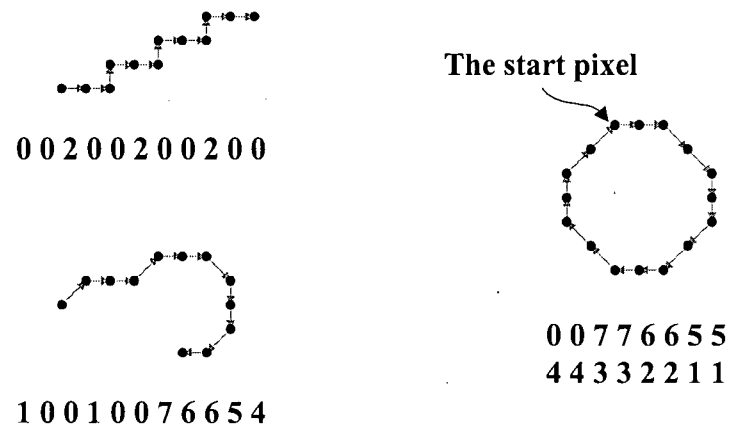
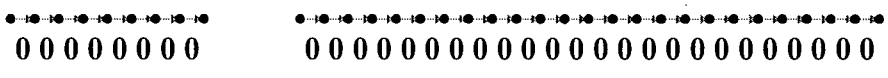


Figure 4.4 Examples of edges and their chain codes.

4.2 Edge expansion process

Edges in the edge image are classified into one of two types:

1) Straight line edges i.e., horizontal lines, vertical lines, or lines with 45° or 135° slope values. The chain code of such a line has a series of the same chain code number e.g. 3, 3, ..., 3. These types of edges are expanded by a straightforward technique, which is that of repeating the chain code of the edge as many times as the expansion factor. For example consider an edge with chain code [0 0 0 0 0 0 0 0], which is a total of 8 pixels at the angle 0°. Assume this horizontal line edge is to be expanded 3 times, then the expanded result of that edge is a total of 24 pixels at the angle 0° i.e. a horizontal line edge also [79]. **Figure 4.5** shows this edge and its expanded edge.



An original horizontal edge **The expanded horizontal edge**

Figure 4.5 Example of 8 pixels horizontal line edge and its expanded edge.

2) Slope line edges other than those in (1) above. These edges are simply formed of two or more short connected lines that are shifted from each other by one pixel in the horizontal, vertical, or diagonal direction depending on the values of their slope. **Figure 4.6** shows examples of slope edges in the edge image.

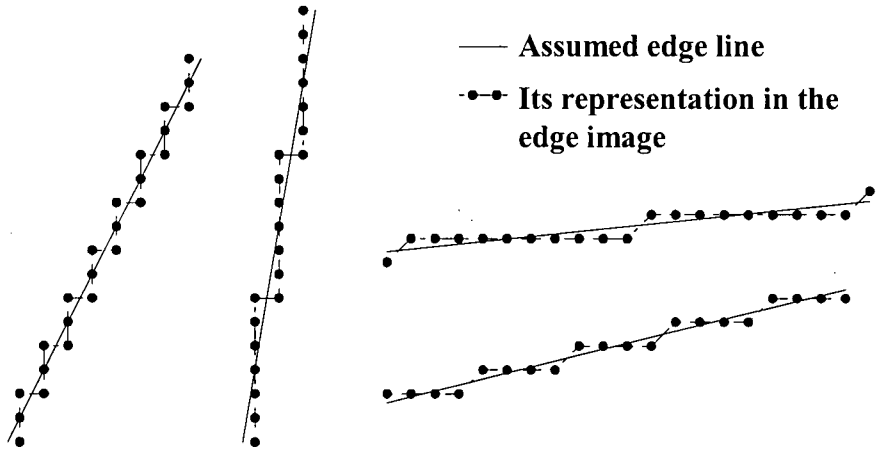
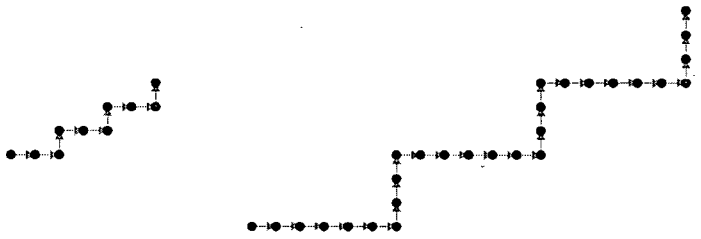


Figure 4.6 Examples of slope line edges.

The chain code of a slope line edge consists of a series of the same chain code with a single change in the chain code (CCC) between pixels, which means the pixels have the same direction but with a single change in direction between pixels, which represents the single shift. For example consider the slope line edge with the chain code [2 2 1 2 2 2], in this example the slope line edge is a series of 2's and the single change is 1, i.e. the change in the chain code (CCC) is 1. This edge is a line with six pixels long at the angle 76°.

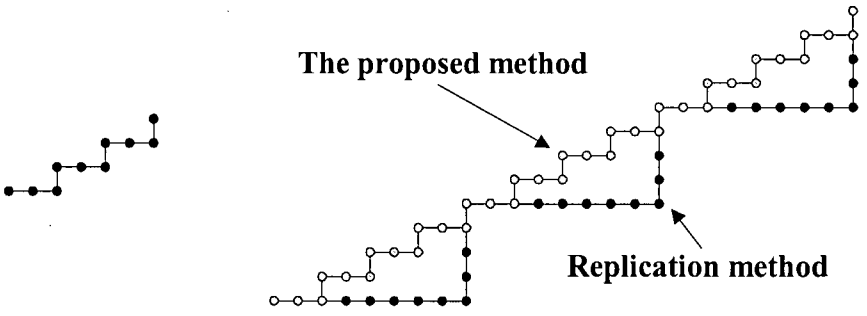
Expanding a slope line edge by repeating the chain code of every pixel by the expansion factor number is called edge-shape-replication and introduces a staircase phenomenon that causes the unnatural look of the edges in the expanded edge image as shown in **Figure 4.7**. The higher the expansion factor the more unnatural and annoying the staircase phenomenon is.



(a) (b)

Figure 4.7 (a) A slope line edge, (b) Its expanded result by repeating the chain code 3 times, (expansion factor = 3).

Obviously, using the edge-shape-replication method does not enlarge slope line edges correctly. The proposed method described below enlarges the slope line edges efficiently as shown in **Figure 4.8**.



(a) (b)

Figure 4.8 (a) A slope-line edge, (b) Its expanded results using edge-shape-replication and the proposed methods, (expansion factor = 3).

4.3 The Proposed Method

After representing all the edges in the edge image as chain code sequences, the slope line edges are then found in these sequences. A slope line edge was defined earlier as a series of the same chain code number (same direction) with single change between pixels, which is called a change in chain code (CCC). **Figure 4.9** shows an example of a slope line edge with the chain code [0 0 0 2 0 0 0 2 0 0 0 2 0 0 0], which has changes in its chain code i.e. 3 CCCs, where the slope line edge is a series of 0's and the CCCs are equal to 2.

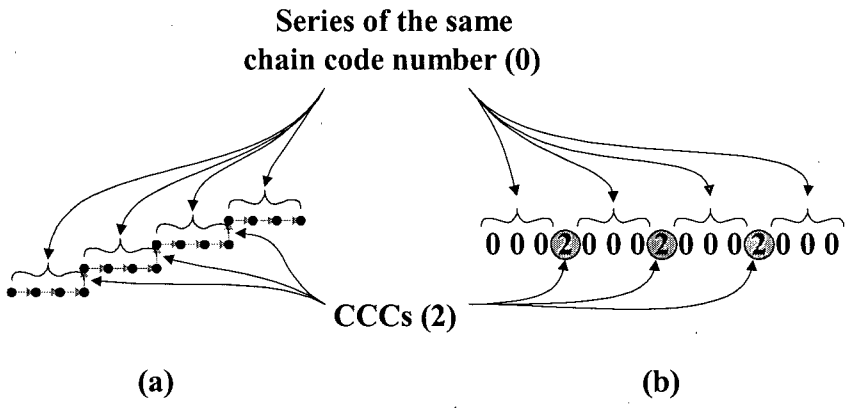


Figure 4.9 An Example of a slope line edge, (a) Actual edge in the edge image, (b) Its chain code representation.

We notice that to recognize an edge as a slope line edge simply involves checking the CCC's in the chain code. The CCC detector starts from the second number in the chain code – because the first number cannot be a CCC – and checks all the numbers of the chain code. A chain code number is considered a CCC only if the number before it and the number after it are equal and both are different from its value as shown in **Figure 4.10**.

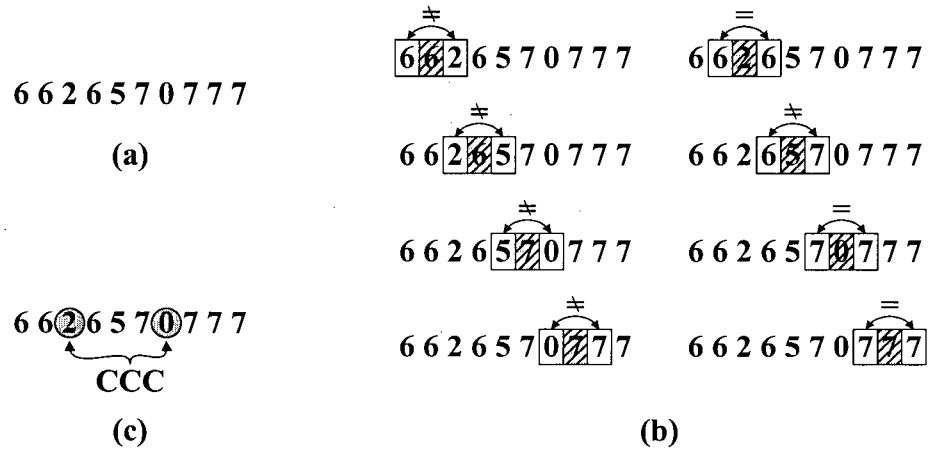


Figure 4.10 Example of the CCC detector, (a) The chain code, (b) The detection processes, and (c) The CCC detector result where only two CCCs are detected.

The series of the same chain code numbers before the CCC, and the series of the same chain code numbers after it forms one series, which is called a segment. The series before is called pre-CCC, and the series after is called post-CCC. For example, consider the segment with the chain code [0 0 0 2 0 0 0], which is part of a slope line edge. Clearly, the 2 is the CCC, and the series of three 0's before it is the pre-CCC and the series of the three 0's after it is the post-CCC as shown in **Figure 4.11**.

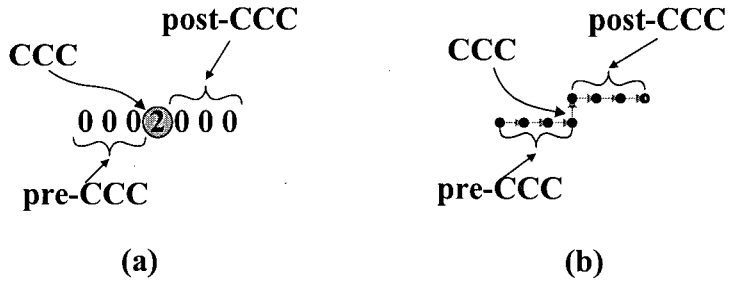


Figure 4.11. Example of a segment of a slope line edge, its pre-part, the CCC, and its post-part, (a) The segment chain code, and (b) The edge in the edge image.

A long slope-line edge consists of more than one CCC in its chain code sequence. In this case, segments are overlapped, as a pre-CCC of a segment is a post-CCC of another as shown in **Figure 4.12**.

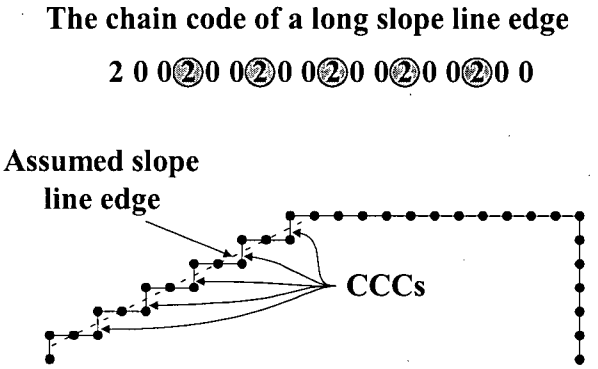


Figure 4.12 A long slope line edge and its chain code.

All slope line edges can be separated into segments whose properties are listed in a table that lists the edge position, edge direction, CCC direction, and the length of the pre-CCC and post-CCC of a segment. **Table 4.1** shows an example of the edge with slope lines shown in **Figure 4.13**, whose chain code is [0 7 6 7 7 6 7 7 6 5 5 6 5 6 5 6 6 6 6 6 7 6 6 6 6 6 6 6 6 6 6 6 6 6 4 6 6].

Table 1 Sample of the segments' properties table.

Pre-part Direction	Change in Chain Code (CCC)	Position in chain code	Pre-part Length	Post-part length
7	6	3	1	2
7	6	6	2	2
5	6	12	2	1
5	6	14	1	1
6	7	22	5	15
6	4	38	15	2

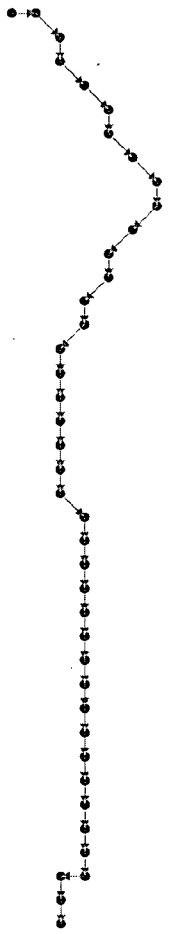


Figure 4.13 Example of the edge that is analyzed in Table 4.1

Before expanding (enlarging) a slope line edge, its edge position must be estimated. We consider two estimated positions only: (1) completely above the slope line edge, and (2) completely under the slope line edge. **Figure 4.14** shows these two estimated positions. The first position is considered in the proposed algorithm.

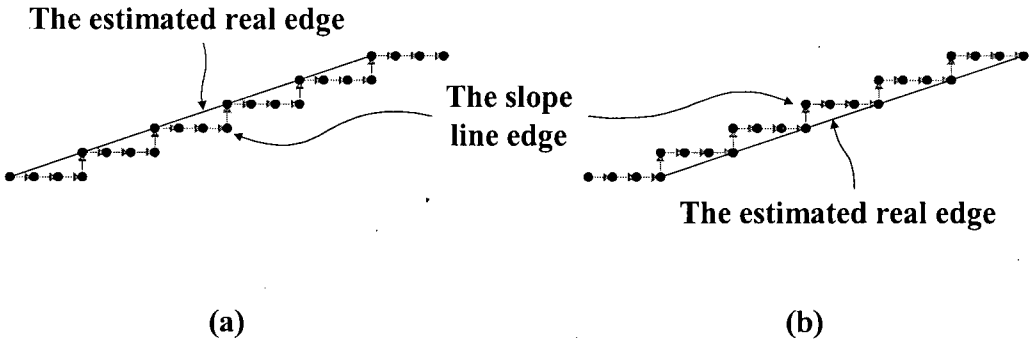


Figure 4.14 The two estimations of the real edge, (a) Position 1 (above), and (b) Position 2 (under).

In the first case, the edge is estimated to be above the slope line edge. Here the area under the estimated edge position is formed of triangles whose two sides are the CCC pixel and its pre-CCC as shown in **Figure 4.15 (a)**. The chain code sequence of that example is [0 0 0 2 0 0 0 2 0 0 0]. In **Figure 4.15 (b)** this edge is separated into

1) A triangle [0 0 0 2], which consists of the pre-CCC [0 0 0] and the CCC [2],
2) A triangle [0 0 0 2], which consists of the pre-CCC [0 0 0] and the CCC [2] ,
3) A line [0 0 0], which is the post-CCC.

Enlarging a slope line edge using the traditional replication method introduces blockiness effects and the chain code sequence when the expansion factor is equal to 4 will be [0 0 0 0 0 0 0 0 0 2 2 2 2 0 0 0 0 0 0 0 0 0 2 2 2 2 0 0 0 0 0 0 0 0 0 0 0 0] as shown in **Figure 4.15 (c)**. Our proposed method regenerates the estimation of the slope line edge, using the same original triangle sequence [0 0 0 2], four times (the expansion factor = 4 in this example), the enlarged triangle sequence then becomes [0 0 0 2 0 0 0 2 0 0 0 2 0 0 0 2], as shown in **Figure 4.15 (d)**. Repeating the same process for every triangle produces the enlarged slope line edge efficiently, and the final enlarged edge

chain code sequence is [0 0 0 2 0 0 0 2 0 0 0 2 0 0 0 2 0 0 0 2 0 0 0 2 0 0 0 2 0 0 0 2 0 0 0 2 0 0 0 2 0 0 0 2], as shown in **Figure 4.15 (e)**.

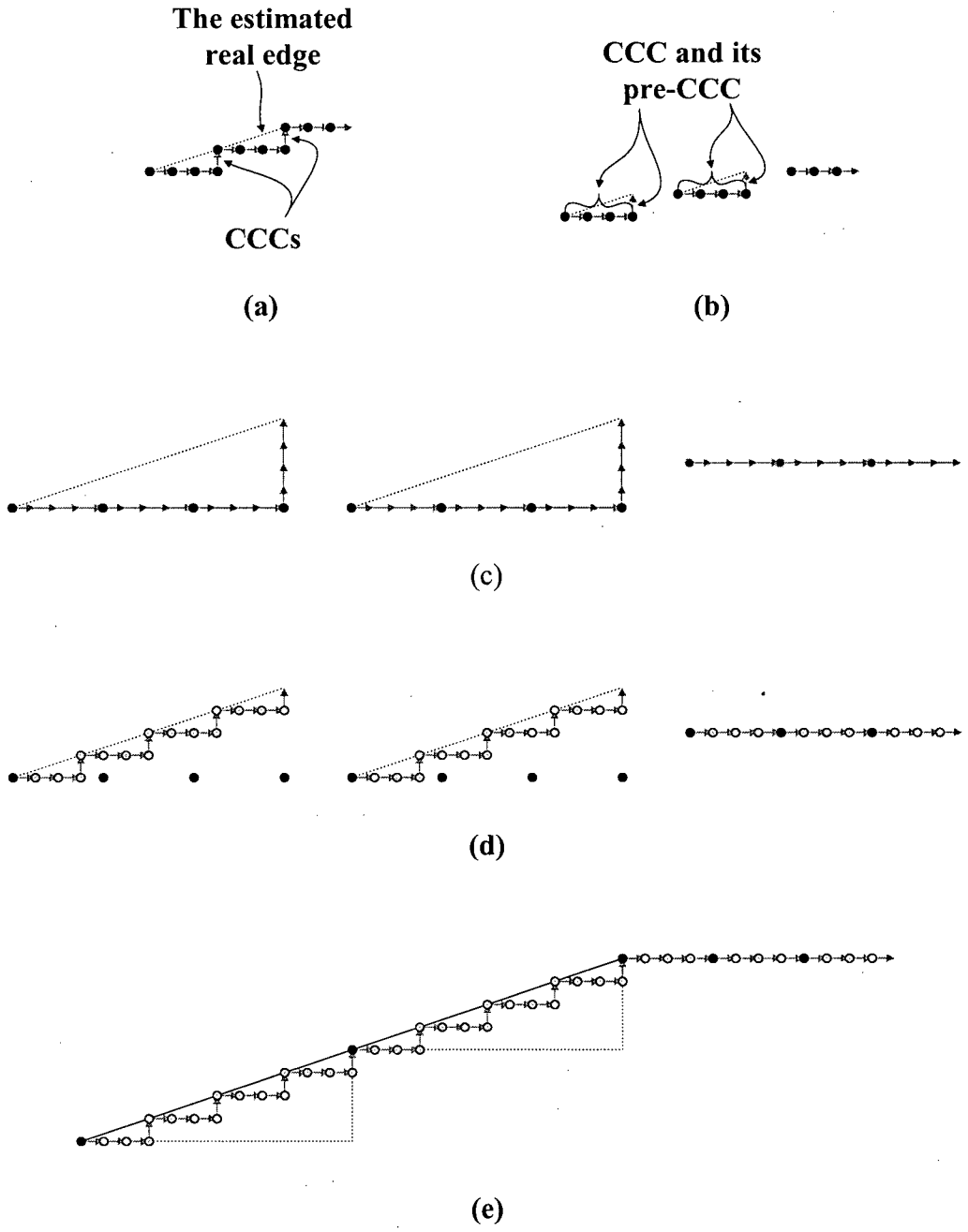


Figure 4.15 The expansion process of the first estimated position which is above the slope line edge.

In the second case, the edge is estimated to be under the slope line edge, the area above the estimated edge position is simply formed of triangles. Two sides of the triangle are formed by the CCC pixel and its post-CCC as shown the example in **Figure 4.16 (a)**. The chain code sequence of this example is [0 0 0 2 0 0 0 2 0 0 0]. In **Figure 4.16 (b)** the edge is separated into

1) A line [0 0 0], which is the pre-CCC,
2) A triangle [2 0 0 0], which consists of the CCC [2] and the post-CCC [0 0 0],
3) A triangle [2 0 0 0], which consists of the CCC [2] and the post-CCC [0 0 0].

Expanding a slope line edge using the traditional replication method introduces blockiness effects and the chain code sequence when the expansion factor is 4 will be [0 0 0 0 0 0 0 0 0 0 0 0 2 2 2 2 0 0 0 0 0 0 0 0 0 0 0 0 2 2 2 2 0 0 0 0 0 0 0 0 0 0 0 0] as shown in **Figure 4.16 (c)**. Our proposed method regenerates the estimation of the slope line edge, using the same original triangle sequence [2 0 0 0], (four times, the expansion factor = 4 in this example), the enlarged triangle becomes [2 0 0 0 2 0 0 0 2 0 0 0 2 0 0 0], as shown in **Figure 4.16 (d)**. Repeating the same process for every triangle above the real edge the expansion produces the enlarged slope line edge efficiently, and the final enlarged edge chain code sequence is [0 0 0 0 0 0 0 0 0 0 0 0 2 0 0 0 2 0 0 0 2 0 0 0 2 0 0 0 2 0 0 0 2 0 0 0 2 0 0 0 2 0 0 0 2 0 0 0], as shown in **Figure 4.16 (e)**.

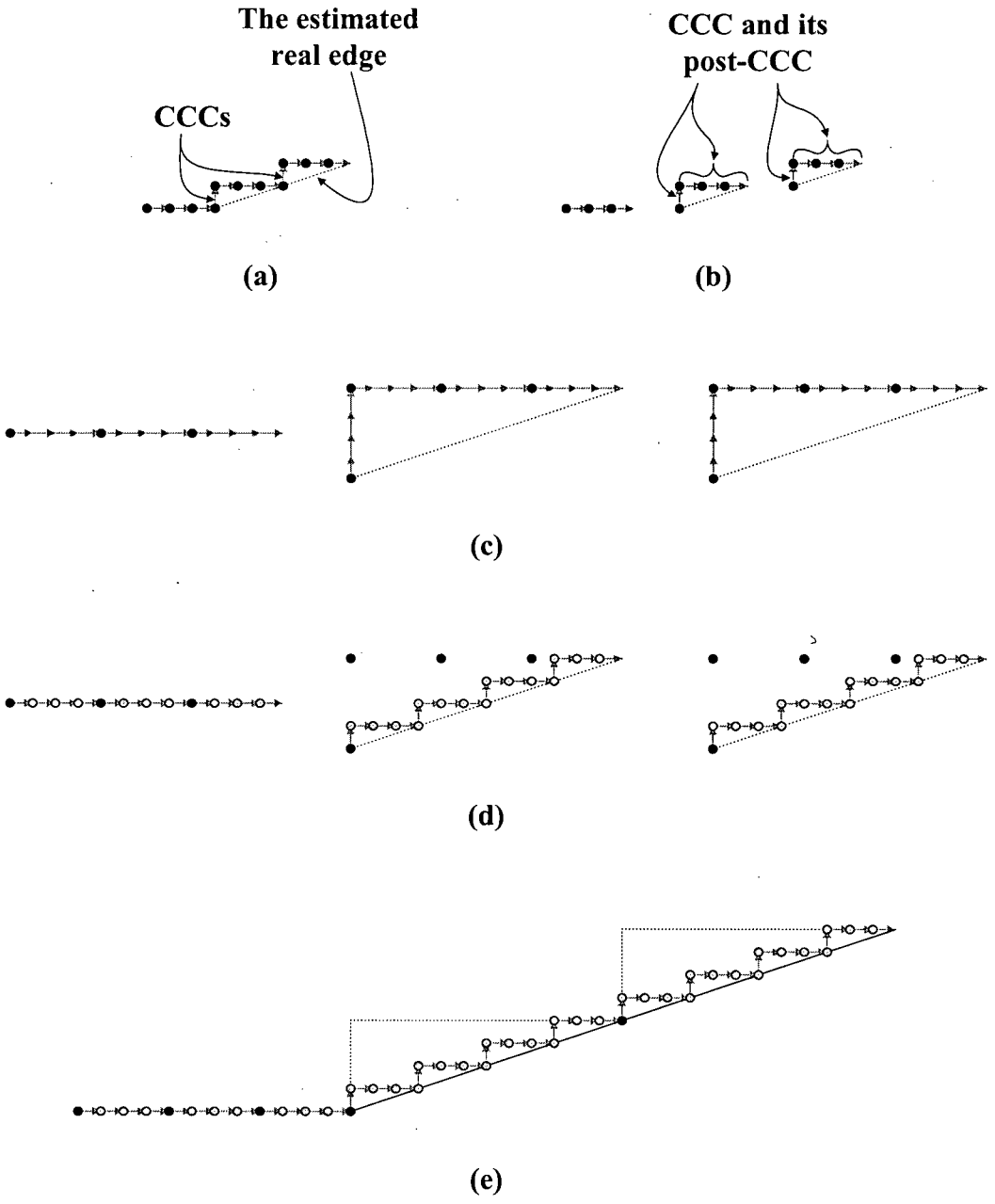


Figure 4.16 The expansion process of the second estimated position which is under the slope line edge.

4.4 Slope Line Edges Cases

There are 2 different cases of slope line edges. This classification depends on the length of the segment after the change in the chain code i.e. the post-CCC of the segment relative to the length of its pre-CCC. In general, the length of the pre-CCC of a segment is close to the length of its post-CCC in value. This is the first case, and is considered to be the short post-CCC case. The other case is the long post-CCC case, where the post-CCC is considered to be long i.e. it is at least twice as long as the pre-part of the same segment. For example, consider a segment whose pre-CCC length is 3 pixels, it will be considered a short post-CCC segment if the post-CCC length is 5 pixels or less. On the other hand, it will be considered a long post-CCC segment if the post-CCC length is 6 pixels or more. These two cases are expanded differently [80, 81].

Case 1 (short post-CCC segment):

In this case, the post-CCC length is greater or equal to twice the pre-CCC length. For example, consider the slope line edge with chain code [X 0 0 2 0 0 0 Y], where the 2 is the CCC, and the segment is [0 0 2 0 0 0]. The pre-CCC length is 2 pixels, and the length of the post-CCC is 3 pixel, which is obviously less than 4. In this work, we consider the estimated edge position to be position 2, which is the under the slope line edge. Position 1 could also be taken but then the positions and the processes for the other cases should be reversed. **Figure 4.17** shows this example. As described before, repeating the triangle that is made of the CCC and the post-CCC of the segment N times, where N is the expansion factor expands this type of segments.

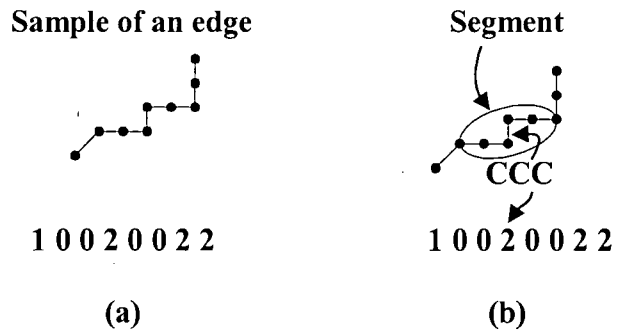


Figure 4.17 Short post-CCC slope line edge (case 1), (a) The slope line edge and its chain code, (b) the segment, its pre- and post-CCCs, and the CCC.

Figure 4.18 shows the expansion process. In general, a slope line edge is made of few overlapped segments. To expand such an edge, the triangle of the CCC and the post-CCC of every segment is repeated N times, where N is the expansion factor. **Figure 4.19** shows an example of a slope line edge with few overlapped segments.

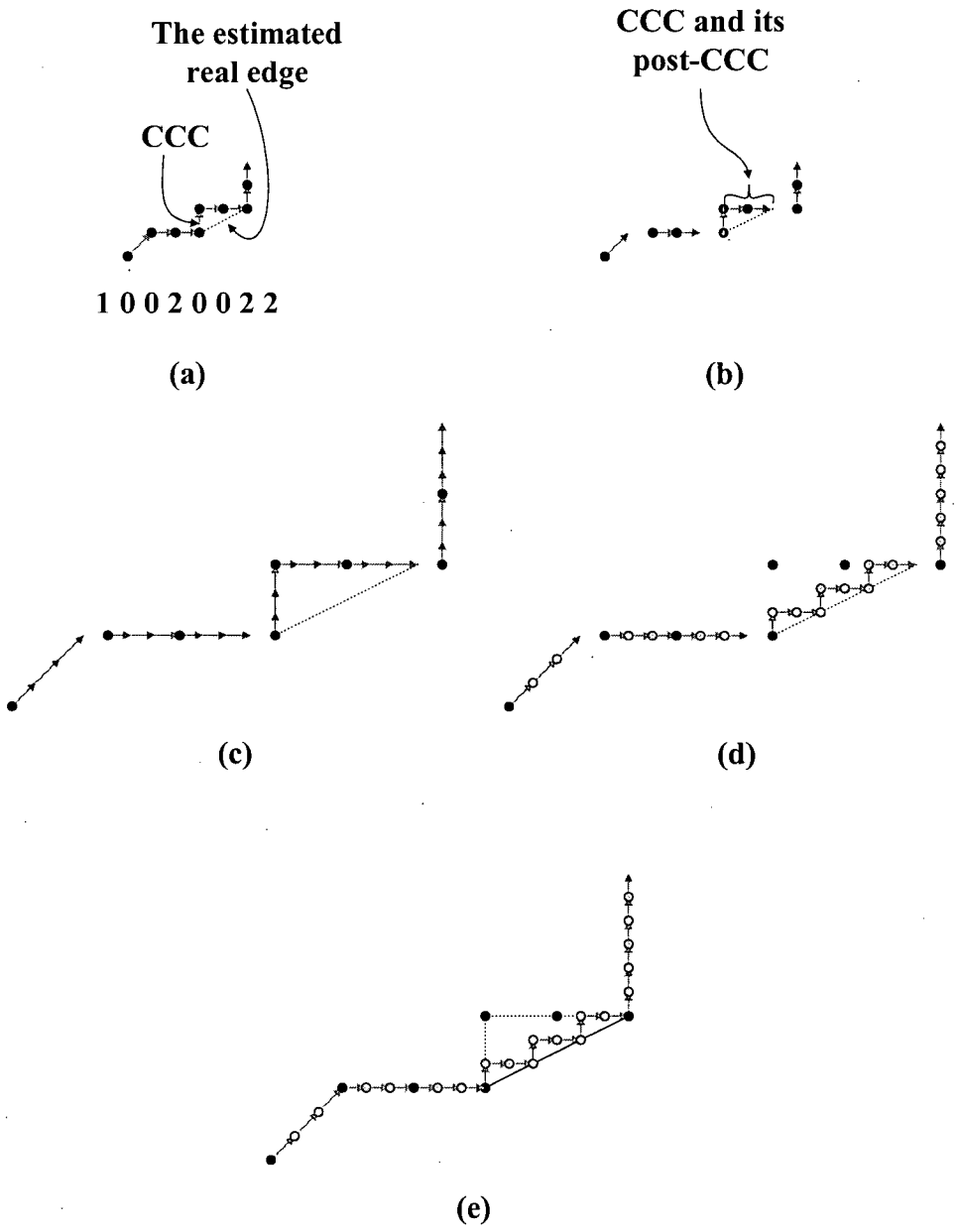
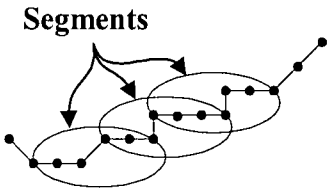
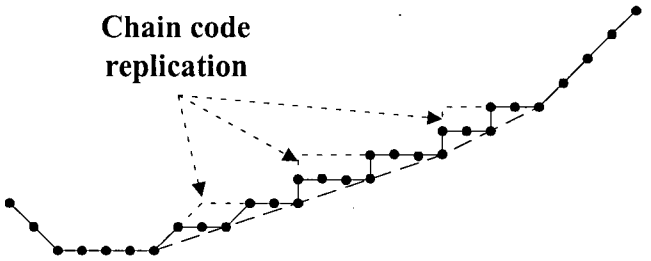


Figure 4.18 Short post-CCC slope line edge (case 1), (a) The slope line edge, (b) The sample of an edge divided into pieces, the pieces expanded by (c) The replication method, (d) The proposed method, and (e) The expanded edge by using the proposed method.



7 0 0 1 0 0 2 0 0 0 2 0 0 1 1

(a)



7 7 0 0 0 0 1 0 0 1 0 0 2 0 0 0 2 0 0 0 2 0 0 2 0 0 1 1 1 1

(b)

Figure 4.19 Short post-CCC slope line edge (case 1) with few segment, (a) The slope line edge, (b) The expanded edge by using the proposed method.

Case 2: (long post-CCC segment):

In this case, the post-CCC length is greater or equal to twice the pre-CCC length. For example, consider the slope line edge with chain code [X 0 0 2 0 0 0 0 0 0 Y], where the 2 is the CCC, and the segment is [0 0 2 0 0 0 0 0 0]. The pre-CCC length is 2 pixels, and the length of the post-CCC is 6 pixel. In this work, we consider the estimated edge position to be position 2, which is the under the slope line edge. **Figure 4.20** shows this example.

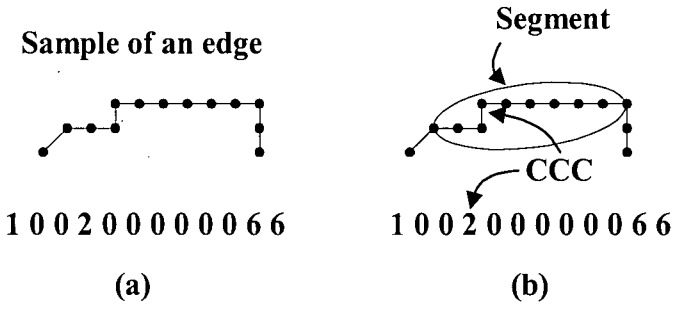


Figure 4.20 Long post-CCC slope line edge (case 2), (a) The slope line edge and its chain code, (b) the segment, its pre- and post-CCCs, and the CCC.

Repeating the triangle, which is made of the CCC and the post-CCC of the segment, does not expand this type of segment correctly. To expand this type of segments correctly, the triangle to be repeated is made of the CCC and only part of the post-CCC. The length of the considered part is equal to the length of the pre-CCC, in the above example, the triangle to be repeated N times, is the CCC and the only two pixels of the post-CCC [2 0 0], where the CCC is 2, and N is the expansion factor. **Figure 4.21** shows the expansion process. **Figure 4.22** shows an example of a slope line edge with few overlapped segments and the last segment is a long one.

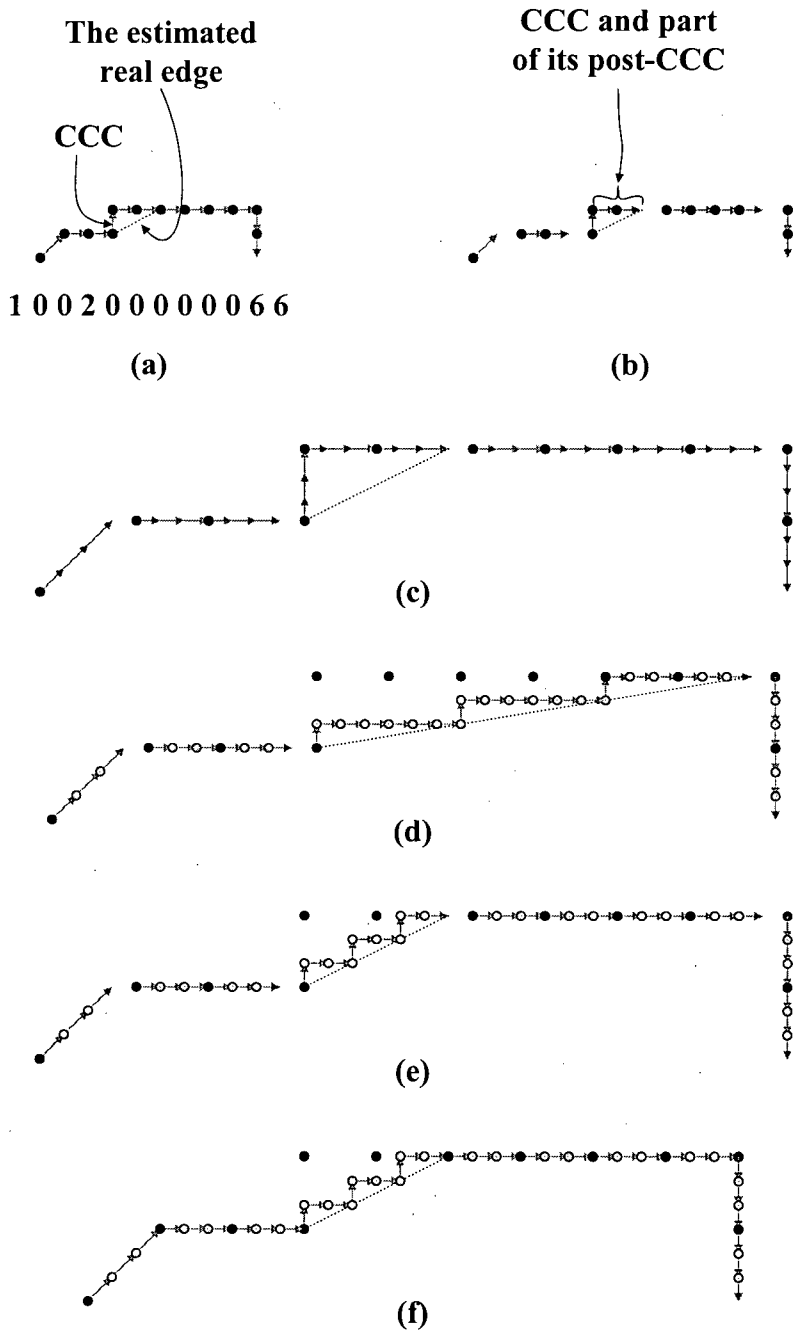
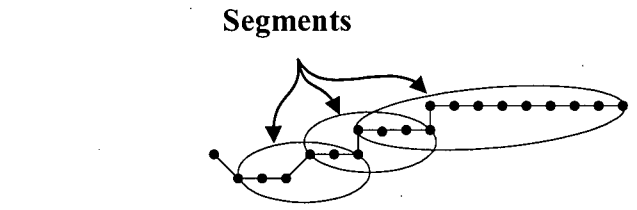
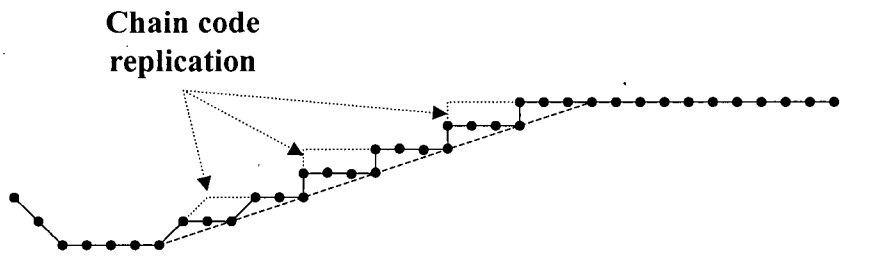


Figure 4.21 Long post-part slope line edge (case 2), (a) The slope line edge, (b) The sample of an edge divided into pieces, the pieces expanded by (c) The replication method, (d) Expanding the segment as a case 1 segment, (e) The proposed method, and (f) The expanded edge by using the proposed method.



7 0 0 1 0 0 2 0 0 0 2 0 0 0 0 0 0 0

(a)



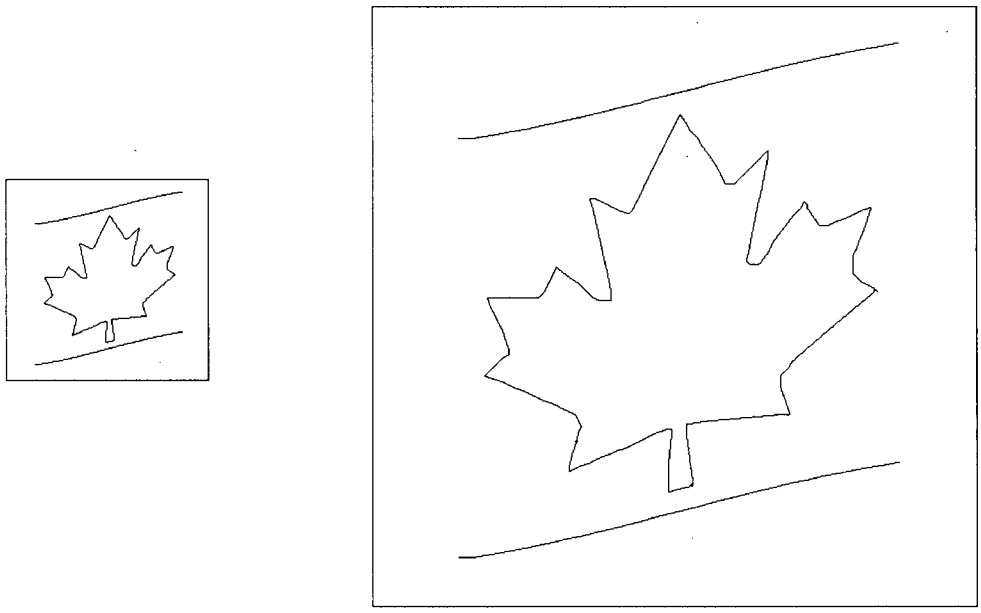
7 7 0 0 0 0 1 0 0 1 0 0 2 0 0 0 2 0 0 0 2 0 0 0 2 0 0 0 0 0 0 0 0 0 0 0 0 0 0

(b)

Figure 4.22 Slope line edge (case 2), (a) The slope line edge and its segment, (b) the edge estimated position, and (c) The 3 times expanded result.

4.5 Examples

The result of edge image expansion is obviously clear in an image that contains only lines such as "Maple Leaf" and "Texas" images which are shown in **Figure 4.23** and **Figure 4.24**.



(a) (b)

Figure 4.23 The "Maple Leaf" image (a) The original edge image, (b) The expanded edge image using the proposed method, the expansion factor is 3.

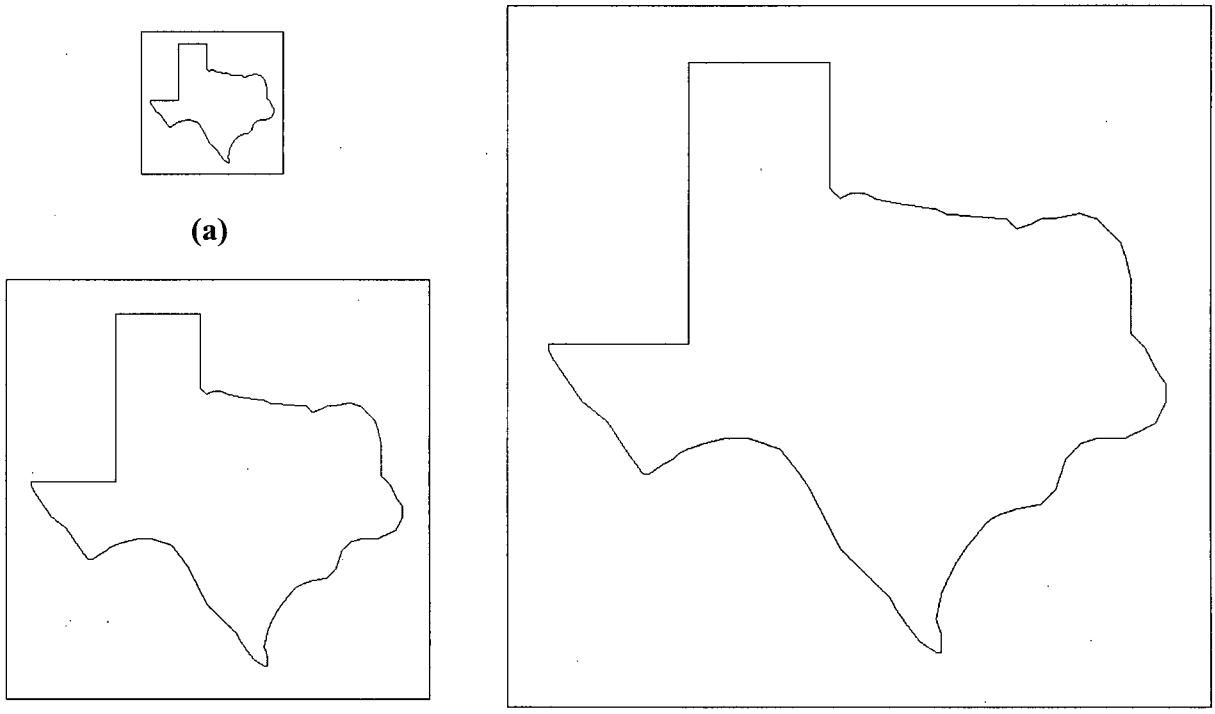


Figure 4.24 The "Texas Map" image (a) The original edge image, and the expanded images using the proposed method, the expansion factors are (b) 3, and (c) 5.

4.6 Binary Images Expansion

Laplacian edge detector produces closed edges, which means closed regions. Expanding a closed edge produces a closed expanded edge. Flood filling is used to fill the closed expanded regions. **Figure 4.25** shows an example of expanding "ABC" binary image using three different conventional methods and the proposed method. In that example the original image reduced by 2, then expanded by 2 using the same method for reduction then expansion. Another example is the "Rabab" image, which is the Arabic writing of the name Rabab. **Figure 4.26** shows the reduction by 2 then expansion by 2 using three conventional methods and the proposed method.

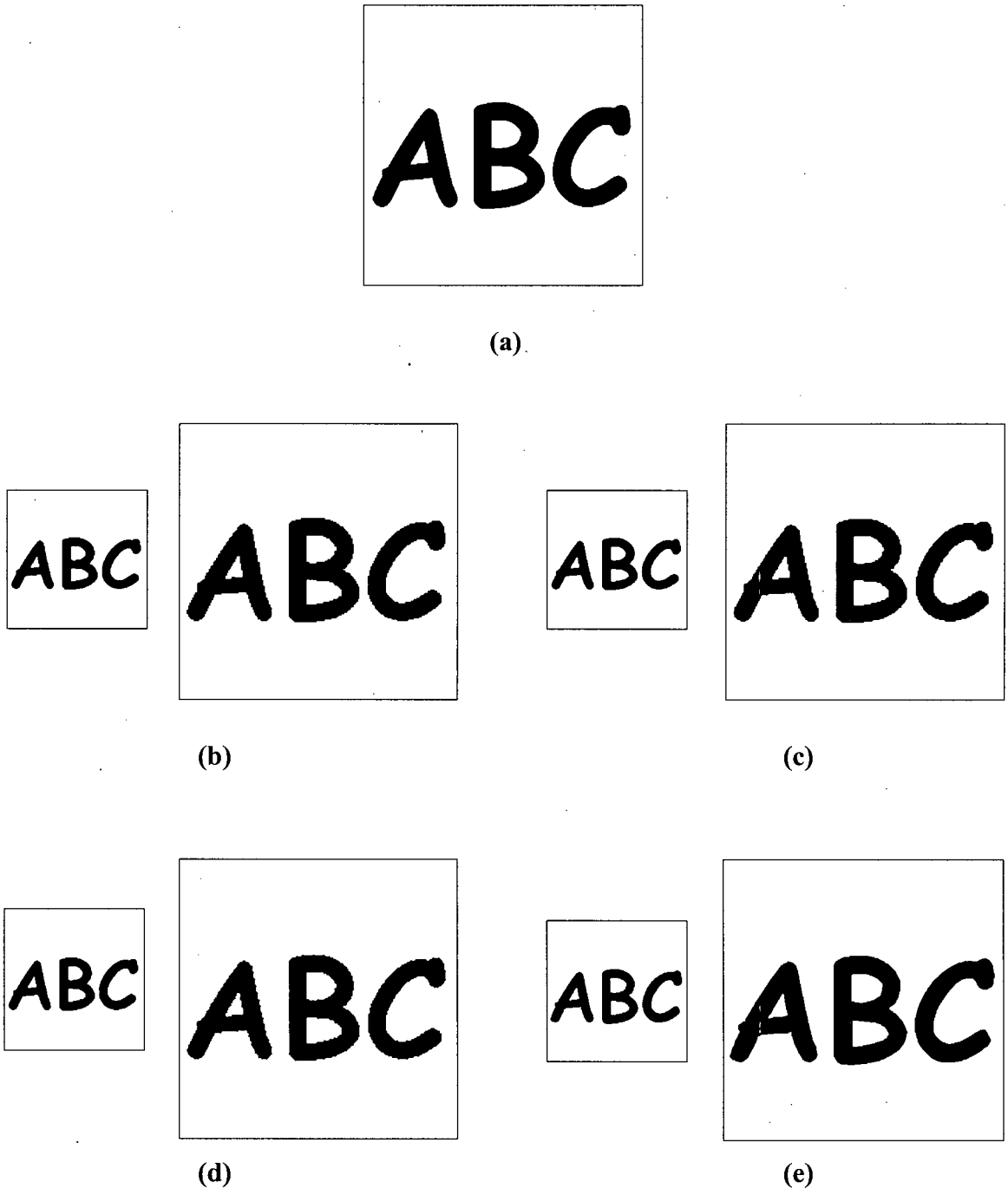


Figure 4.25 The "ABC" Image. (a) The original binary image, and its reduced then expanded by 2 images using four different methods (b) Replication, (c) Linear Interpolation, (d) Cubic Interpolation, and (e) The proposed method.

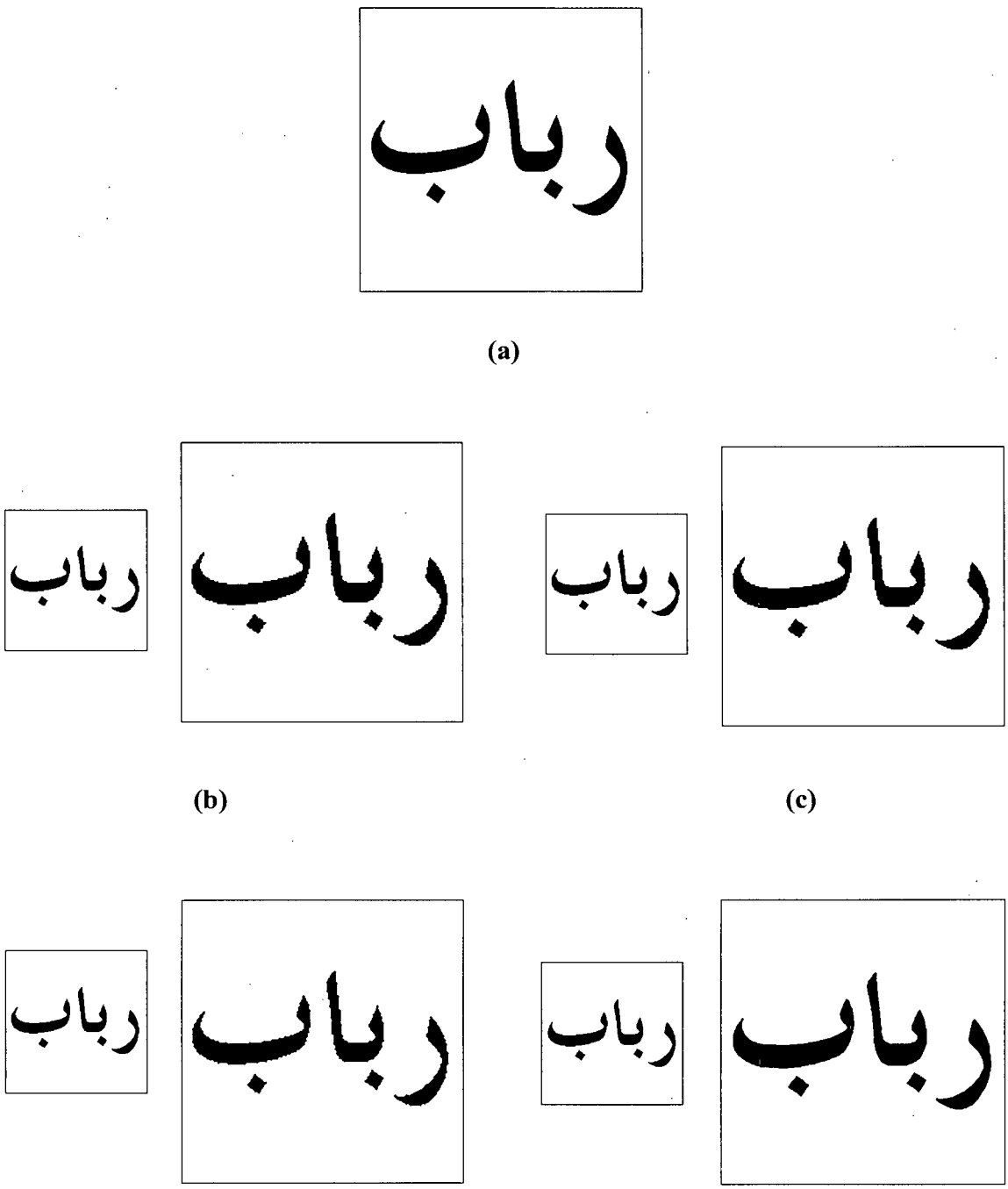


Figure 4.26 The "Rabab" Image. (a) The original binary image, and its reduced then expanded by 2 images using four different methods (b) Replication, (c) Linear Interpolation, (d) Cubic Interpolation, and (e) The proposed method.

4.7 Gray-Level Image Expansion

The Canny edge detector [67] is used to produce the edge image of a gray-level image. The Canny edge detector produces single width pixel edges as the edge image [82]. Edges in the edge image are coded and expanded following the same procedure that is used with the binary image expansion and which is discussed in the previous section [82]. However, expanding the inside or the homogenous regions of gray-level images is different from that of the binary images.

Expansion process

All original pixels are mapped onto the expanded image. The expansion procedure introduces many new pixels in the expanded image, which have to be filled. For example, expanding an image 2 times introduces three new pixels for every original pixel as shown in **Figure 4.27**.

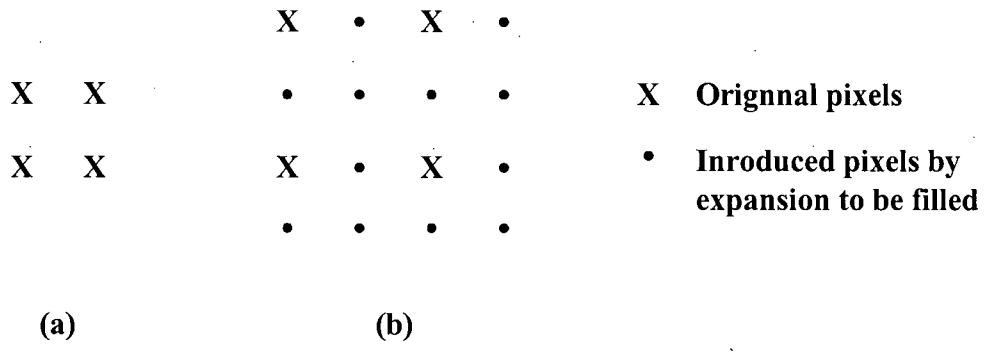


Figure 4.27 Example on the original pixels mapping onto the expanded image (a) The original pixels, (b) The expanded image.

The introduced pixels are filled differently depending on their position relative to an edge. There are three different kind of pixels depending on their positions relative of an

edge: (a) inside pixel (far from an edge), (b) edge pixel (lies on the edge itself), and (c) near edge pixel (within two pixels distance from an edge), as shown in **Figure 4.28**.

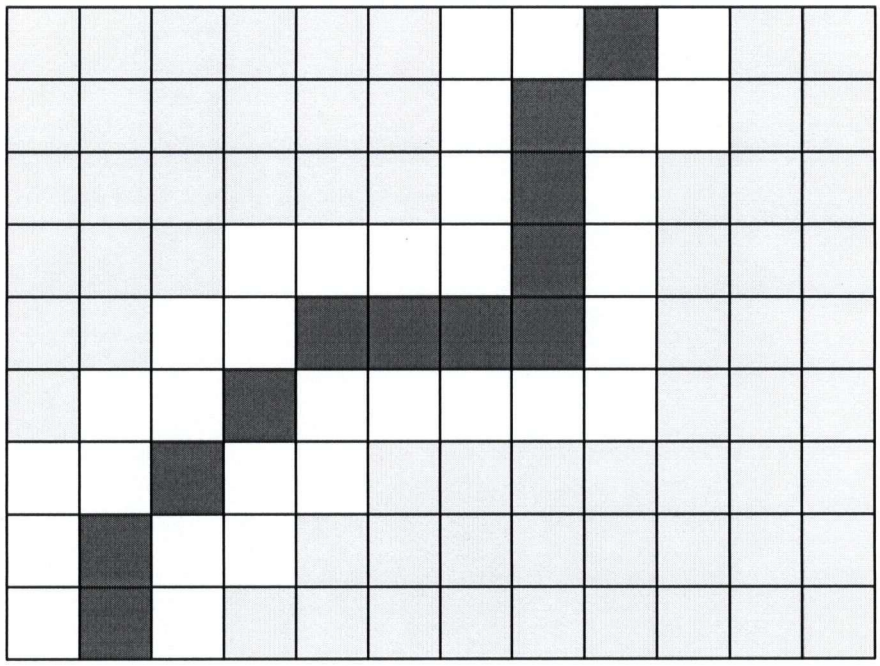


Figure 4.28 The three different pixel position with respect to an edge. Gray pixels lie in a homogenous region, Dark pixels lie on an edge, White pixels lie near edge pixels.

(a) Inside pixels: a window of 3x3 is used to determine if the pixel to be expanded is an inside pixel or not. This type includes all pixels that are 2 or more pixels away from an edge and are considered to form pixels in homogenous regions. These pixels are filled using adaptive averaging filters. Inside a homogenous region, the introduced pixel position is one of three different positions with respect to the original pixel. The three different positions are to the right, to the bottom and to the right-bottom of the original pixel as shown in **Figure 4.29**. Each of these positions is expanded with a different window. The window coefficients (a and b) determine the method that will be used to fill

the new pixel. Giving a and b the values 1 and 0, respectively, fides the value of the new pixel using the replication method. Giving them both the same values (i.e. in this case 0.5), finds the value of the new pixel using the linear interpolation method. The proposed method uses the value 0.7 for a and the value 0.3 for b.

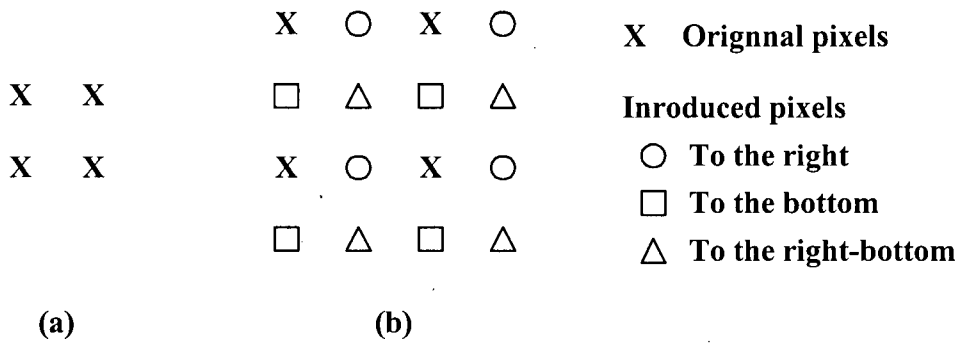


Figure 4.29 (a) The original pixels. (b) The three different positions of the inside region pixels.

A pixel to the right of the original pixel lies horizontally between two original pixels and is represented as a circle in **Figure 4.29**. Such pixels are filled using the window that is shown in **Figure 4.30**.

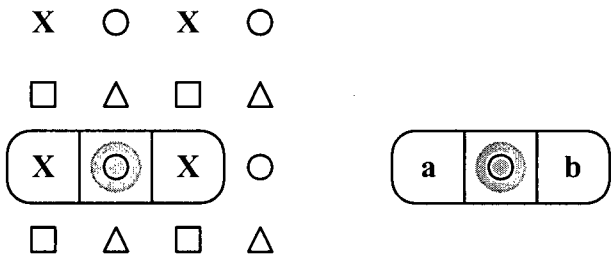


Figure 4.30 The window that calculates the value of the pixels to the right of an original pixel.

A pixel to the bottom of the original pixel lies vertically between two original pixels and is represented as a square in **Figure 4.29**. Such pixels are filled by also using the averaging filter as shown in **Figure 4.31**. The window coefficients are 0.7 for a and 0.3 for b, as explained before.

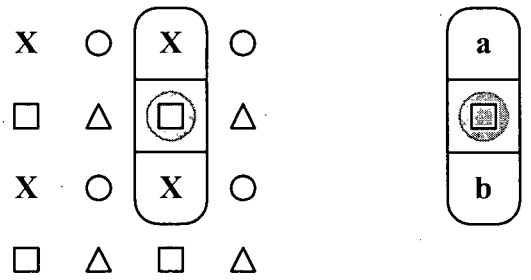


Figure 4.31 The window that calculates the value of the pixels to the bottom of an original pixel.

A pixel to the right-bottom of the original pixel lies in the middle among four original pixels and is represented as a triangle in **Figure 4.29**. Such pixels are filled using the averaging filter shown in **Figure 4.32**. The window coefficients can be chosen as explained before. The window coefficients are 0.7 for a and 0.1 for b, as explained before.

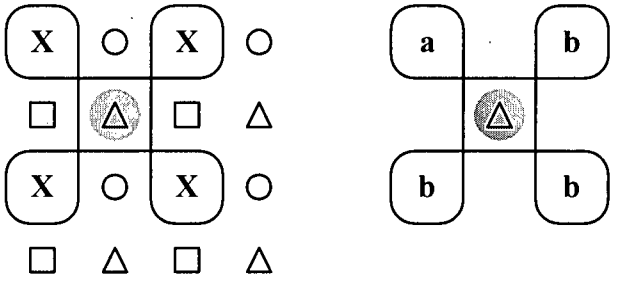


Figure 4.32 The window that calculates the value of the pixels to the right-bottom of an original pixel.

(b) Edge pixels: are pixels that lie on an edge itself. As discussed earlier the shape of the edge such as in **Figure 4.33 (a)**, whose chain code is represented by arrows in **Figure 4.33 (b)**, would be expanded to become as shown in **Figure 4.33 (c)**. However, repeating these pixels' values as shown in **Figure 4.33 (d)**, produces a zigzag effect, as shown in **Figure 4.33 (e)**.

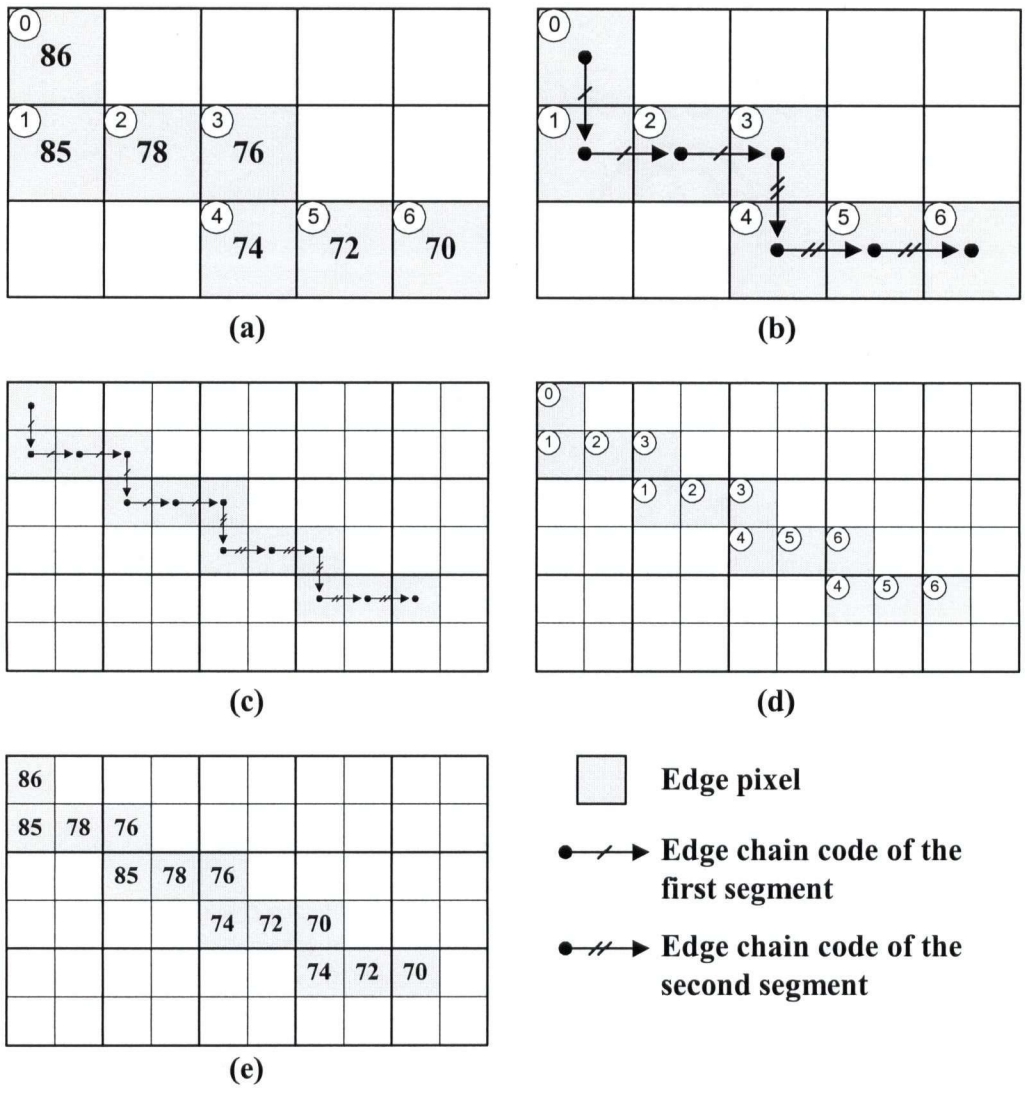


Figure 4.33 Example of filling the edge pixels by repeating the original edge pixels' values. (a) The original edge pixels, (b) The chain code represented by arrows, (c) The expanded edge, (d) The values of the expanded edge pixels repeated from the original pixels, and (e) The edge pixels values.

The proposed method suggests that any original edge pixel that does not lie on an expanded edge, its value will be altered, as shown in **Figure 4.34 (b)**, and it will be calculated the next step with the next type of pixels **(c)**. The value of a new edge pixel is calculated from its two nearest original edge pixels, which are original pixels and lie on the expanded edge too. The linear interpolation technique is used to calculate the value of a new pixel from its nearest two known edge pixels. For example in Figure 4.34 (b), the unknown value of the pixel that lies between the 78 and the 76 pixels is 77.

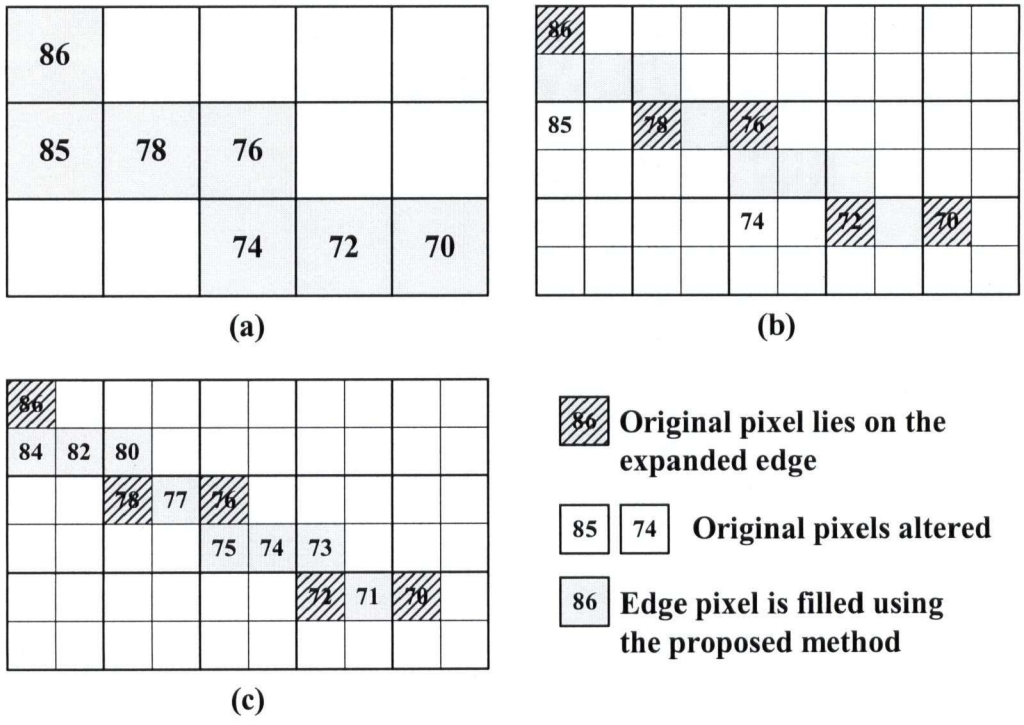


Figure 4.34 Example of filling the edge pixels by the proposed method. (a) The original edge pixels, (b) The original edge values that were kept or altered for the expanded edge, and (c) The calculated values of the expanded edge.

(c) Near edge pixels: A near edge pixel is a pixel whose distance from any edge pixel is less than two pixels, where the distance between any 2 horizontal, vertical or diagonal pixels is considered to be equal to 1. These pixels lie in the narrow areas between the homogenous regions and edges and are shown as white squares in **Figure 4.35**. These pixels are computed by averaging their neighbor pixels within the same region, which means the same side of the edge. At most three of the eight neighbor pixels of the considered pixel are averaged depending on the position of the neighbor pixels relative to the edge. The pixels that are averaged are the farthest pixels from the edge. **Figure 4.35** shows examples of filling near edge pixels.

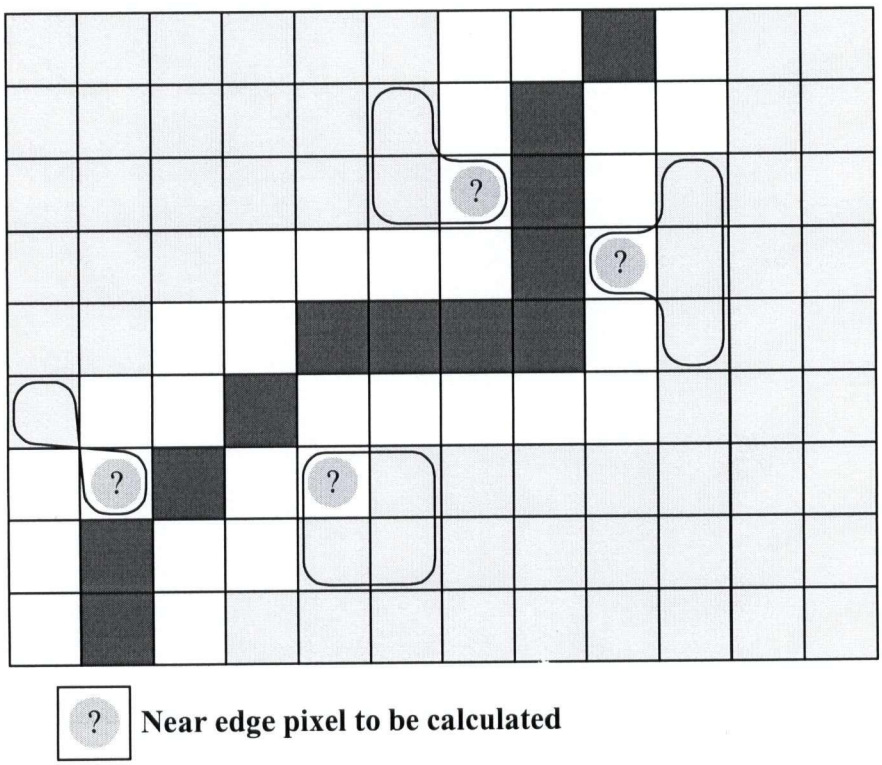


Figure 4.35 Examples of filling near edge pixels. Gray pixels lie in a homogenous region, Dark pixels lie on edge, White pixels lie near edge pixels. The closed shapes are the pixels to be averaged so as to calculate the pixel marked as question mark.

4.8 Results

The proposed method is tested with binary and gray-level images. The result in **Table 2.** is obtained by reducing the original image by factor 2 then expanding them by factor 2 using different methods. The images are reduced by eliminating every other row then eliminating every other column. The sum of absolute error between the original image and the expanded image is used to compare the performance of the replication, linear and cubic interpolation, with the performance of the proposed method. **Table 2.** can show that the proposed method outperforms the other method with the binary and gray-level images.

Table 2 The Sum of the Absolute Error of four images enlarged using three methods and the proposed method

Sum of the Absolute Error					
Image		Replication	Linear Interpolation	Cubic Interpolation	Proposed Method
Binary	**ABC**	964	903	1006	572
	Rabab	931	879	972	545
Multi Gray Levels	**Tropic**	207742	123775	115402	71139
	Saad	295097	153533	140711	93811

In terns of complexity, the number of operations required by our binary proposed method is 30% more than that of the replication method. The types of operations here are additions, subtraction, and assigning. Our gray-level proposed method depends on edge detection (Canny). Here the edge detection consumes around 90% of the total operations

complexity. The other 10% of the computations are consumed by edge coding, edge classifying, edge expansion, and homogenous regions filling. These require around 25% of the number of operations required by the cubic interpolation method.

5 Conclusions and Future Suggestions

5.0 Overview

Image expansion is an essential tool for many applications, such as digital photography, satellite imagery, biomedical image processing, and Internet browsing. The expansion process introduces many new pixels to be filled. For example, expanding the image two times produces three new pixels for each original pixel. To fill these new pixels there are: 1) the conventional methods; 2) the adaptive methods, 3) the statistical methods, and 4) the methods that use new concepts such as wavelet and neural networks. Our proposed method lies in the second category, i.e. the adaptive methods category.

The conventional methods expand the whole image using the same process for every pixel. Some give results that are better than others, but all suffer from one or more of the following effects: blockiness, blurring (over smoothing), contouring, ringing, or

oscillation effects. This is because these types of methods are not optimal, as generally they are not designed to minimize a certain function of error or artifacts. They also lack to consider the local characteristics of the image.

The adaptive methods produce better results than the conventional methods as they consider the local characteristics of the image. Generally, the statistical methods are very computationally demanding and expensive as they compute the local characteristics of the image.

5.1 Conclusions

The proposed method preserves the quality of the expanded image and does not distort edges and corners where most of the information resides. The proposed method adaptively expands the image as it expands the edges in different scheme from the homogenous regions. The expanded edges using the proposed method are better in quality and visually look more natural and near exact.

The proposed method is tested with many images: binary and gray-level images. The Sum of Absolute Error (SAE) technique is used to measure the performance of the proposed method in terms of quality. The proposed method outperforms other methods in term of quality as was shown in **Table 2**.

5.2 Suggestions for Future Work

The following are few suggestions for continuing this research.

- Edge Detection is the first step of the proposed method and it is used to classify the image pixels as an edge pixel or not. This is a critical decision as the following steps depend on this decision. The Canny edge detector yields very good results but it is computationally demanding. For further study, other edge detection methods can be tested or developed.
- Edge coding is an important process on which the quality depends. The proposed method uses relatively simple edge coding technique to create an edge chain code. Using more complicated and more accurate edge coding technique is one of the suggestions for future study. There are many techniques for edge coding that can be used to improve the quality of the expanded edges [83, 84, 85].

- In this thesis project, edges are considered to be one of two cases: 1) short post-CCC segment i.e. the segment after the change in the chain code is short, 2) long post-CCC segment i.e. the segment after the change in the chain code is long. We have another classification that consists of 16 cases instead of two and it produces better results but it is not complete. The more the cases the better the expanded edges the better the quality. That means future study could be done on better classification for the edge.

- Another suggestion for future study is the expansion process of the homogenous regions. The expansion process of the pixels inside the homogenous regions can be done based on a statistical information or the texture information of that region.

Bibliography

[1] R. Schafer; L. Rabiner, "A Digital Signal Processing Approach to Interpolation", Proc. of IEEE Conference 1973, Volume 61, pp. 692-702.

[2] S. Rifman, "Digital Rectification of ERTS Multispectral Imagery", Proc. of , 1973, Vol. 1, pp. 1131-1142.

[3] R. Bernstein, "Digital Image Processing of Earth Observation Sensor Data", IBM J. Res. Dev., 1976, 20, pp.40-57.

[4] J. Parker; R. Kenyon; D. Troxel, "Comparison of Interpolation methods for Image Resampling", IEEE Trans. on Medical Imaging, 1983, 2, (1), pp. 31-39.

[5] H.C. Hsieh; W.T. Chang, "Image interpolation by Analogue Circuit", Electronics Letters, Vol. 28, No. 9, April 1992, pp. 867-868.

[6] J. Markel, "FFT pruning", IEEE Trans. on Audio Electroacoust, 1971, 19, pp. 305-311.

[7] T. Smit; M. Smith; S. Nichols, "Efficient Sinc Function Interpolation Technique for Center Padded Data", IEEE Trans. on Acoustic Speech Signal Processing, 1990, 38, (9), pp. 1512-1517.

[8] J. Ratzel, "The Discrete Representation of spatially Continuous Images", Ph.D. Dissertation, Massachusetts Institute of Technology, Cambridge, 1980.

[9] P. Sankar, and L. Ferrari, "Simple Algorithms and Architecture for B-spline Interpolation", IEEE Trans. on Pattern Analysis and Machine Intelligence, 1988, 10, (2), pp. 271-276.

[10] S. Gustafson; T. Rhoadarmer; J. Loomis; G. Little, "Smart Zooming", Proc. on SPIE-Int. Soc. Opt. Eng., 1994, 2238, pp. 170-175.

[11] H. Hou; H. Andrews, "Cubic splines for Image Interpolation and Digital Filtering", IEEE Trans. on Acoustic and Speech Signal Processing, 1978, 26, (6), pp. 508-517.

[12] C. Lee; M. Eden; M. Unser, "Near Optimal Geometric Image Scaling Using Oblique Projection Operators", Proc. of IEEE Acoustic, Speech, and Signal Processing Conference, Atlanta, GA, USA, 7-10 May, 1996.

[13] J. Taber, "Evaluation of Digitally Corrected ERTS Images", Proc. of Third Earth Resources Technology Satellite-1, 1973, Vol. 1, pp. 1838-1843.

[14] R. Keys, "Cubic Convolution Interpolation for Digital ImageProcessing", IEEE Trans. on Acoustic, Speech and Signal Processing, 1981, 29, (6), pp.1153-1160.

[15] M. Unser, A. Aldroubi, M. Eden, "Enlargement or Reduction of Digital Images With Minimum Loss of Information", IEEE Trans. on Image Processing, Vol. 4, No. 3, March 1995, pp. 247-258.

[16] M. Unser, "Splines a Perfect Fit for Signal and Image Processing", IEEE Trans. on Signal Processing, Vol. 16, No. 6, November 1999, pp. 22-38.

[17] V. Algazi; G. Ford; R. Potharlanka, "Directional Interpolation of Images Based on Visual Properties and Rank Order Filtering", Proc. of International Conference on Acoustic, Speech and Signal Processing, ICASSP'91, Toronto, Canada, 1991, Vol. M12.25.

[18] F.G.B. De Natale, G.S. Desoli, D.D. Giusto, and G. Vernazza, "A Spline-Like Scheme for Least-Square Bilinear Interpolation of Images", Proc. of IEEE Acoustic, Speech, and Signal Processing Conference, New York, NY, USA, April 27-30, 1993

[19] L.A. Ferrari; J.H. Park, "An Efficient Spline Basis for Multi-Dimensional Applications: Image interpolation", Proc. of IEEE International Symposium on Circuits and Systems, Honk Kong, June 9-12, 1997.

[20] T. Chen; R. De Figueiredo, "Two-Dimensional Interpolation by Generalized Spline Filters Based on Partial Differential Equation Image Models", IEEE Trans. on Acoustic, Speech and Signal Processing, 1985, 33, (3), pp. 631-642.

[21] G. Chen; R.J.P. de Figueiredo, "A Unified Approach to Optimal Image Interpolation Problems Based on Linear Partial Differential Equation Models", IEEE Trans. on Image Processing, Vol. 2, No. 1, Jan. 1993, pp. 41-49.

[22] T. Ramstad; Y. Wang, "Efficient Image Interpolation Scheme Using Hybrid IIR Nyquist Filters", Optical Eng., 1992, 31, (6), pp. 1277-1283.

[23] R. Constable; I. Kay; M. Smith; R. Henkelman, "High Quality Zoomed MR Images, Using Discrete Fourier Transform Interpolation", J. Comput. Assist. Tomogr., 1989, 13, (1), pp. 179-181.

[24] P.J.S.G. Ferreira, "The Duality of Two Recent Image Interpolation Methods", Proc. of IEEE International Conference on Image Processing, Lausanne, Switzerland, April 16-19,1996.

[25] H. Chen; G.E. Ford, "An FIR Image Interpolation Filter Design Method Based on Properties of Human Vision", Proc. of 1st International Conference on Image Processing, Austin, TX, USA, Nov. 13-16, 1994.

[26] T. Yamada, R. Ishii, "An Algorithm to Magnify Images", Proc. of IEEE International Conference on industrial Technology, ICIT '96, Shanghai, China, Dec. 2-6, 1996.

[27] J.P. Oakley, "A New Method for Image Interpolation and its Applications in Image Analysis", Proc. of Third International Conference on Image Processing and its Applications, Warwick, UK, 1989.

[28] David Kirk, "Graphics Gems III", Academic Press Professional Inc., 1992.

[29] A.M. Darwish, M.S. Bedair, S.I. Shaheen, "Adaptive Resampling Algorithm for Image Zooming", Proc. of IEE Vision, Image and Signal Processing, Aug. 1997.

[30] Y. Wang; S. Mitra, "Edge Preserved Image Zooming", Proc. of European Signal Processing, EURASIP-88, Grenoble, France, 1988, pp.1445-1448.

[31] S. Thurnhofer; M. Lightstone; S. Mitra, "Adaptive Interpolation of Images With Application to Interlaced-to-Progressive Conversion", Proc. of SPIE-Int. Soc. Opt. Eng., 1991, 1460, pp. 614-625.

[32] S.W. Lee; J.K. Paik, "Image Interpolation Using Adaptive Fast B-Spline Filtering", Proc. of IEEE International Conference on Acoustics, Speech, and Signal Processing, Minneapolis, MN, USA, April 27-30, 1993.

[33] K. Jensen; D. Anastassiou, "Spatial Resolution Enhancement of Images Using Nonlinear Interpolation", Proc. of International Conference on Acoustic, Speech, and Signal Processing, ICASSP'90, Albuquerque, NM, 1990,pp. 2045-2048.

[34]K. Jensen; D. Anastassiou, "Subpixel Edge Localization and Interpolation of Still Images", IEEE Trans. on Image Processing, 1995, 4, (3), pp. 285-295.

[35] N. Herofotou, A.N. Venetsanopoulos, "Colour Image Interpolation for High Resolution Acquisition and Display Devices", IEEE Trans. on Consumer Electronics, Vol. 41, No. 4, Nov. 1995, pp. 1118-1126.

[36] B. Zeng; A.N. Venetsanopoulos, "Comparative Study of Several Nonlinear Image Interpolation Schemes", Proc. of SPIE, Vol. 1818, 1992, pp. 21-29.

[37] A. Lehtonen; M. Renfors, "Nonlinear Quineunx Interpolation Filtering", Proc. of SPIE, Vol. 1360, 1990, pp. 135-142.

[38] S.D. Bayrakeri; R.M. Mersereau, "A New Method For Directional Image Interpolation", Proc. of IEEE International Conference on Acoustics, speech, and Signal Processing, Detroit, MI, USA, May 9-12, 1995.

[39] S. Carrato; G. Ramponi; S. Marsi, "A Simple Edge-Sensitive Image Interpolation Filter", Proc. of IEEE International Conference on Image Processing", Lausanne, Switzerland, Sept. 16-19, 1996.

[40] W. Bender; C. Rosenberg, "Image Enhancement Using Non-Linear Sampling", Proc. of SPIE-Int. Soc. Opt. Eng., 1991, pp. 59-70.

[41] S. Thurnhofer; S. Mitra, "Edge-Ehanced Image Zooming", Optical Eng., 1996, 35, (7), pp. 1862-1869.

[42] G.H. Atwood, W.A. Davis, "Image Expansion Using Interpolation and Heuristic Edge Following", Proc. of IEEE International Conference on Image Processing and its Applications, Warwick, UK, 1989.

[43] P.W. Wong; J.P. Allebach, "Convergence of an Iterative Edge Directed Image Interpolation Algorithm", Proc. of IEEE International Symposium on Circuits and Systems, Honk Kong, June 9-12, 1997.

[44] K. Xue; A. Winans; E. Walowit, "An Edge-Restricted Spatial Interpolation Algorithm", J. Electron. Imaging, 1992, 12, pp. 152-161.

[45] D. Geiger, J.E. Kogler, "Scaling Images and Image Features Via the Renormalization Group", Proc. of IEEE Computer Vision and Pattern Recognition, New York, NY, USA, June 15-17, 1993.

[46] R.R. Schultz, R.L. Stevenson, "A Bayesian Approach to Image Expansion for Improved Definition", IEEE Trans. on Image Processing, Vol. 3, No. 3, May, 1994, pp. 233-242.

[47] N. Herodotou; A.N. Venetsanopoulos; L. Onural, "Image Interpolation Using a Simple Gibbs Random Field Model", Proc. of IEEE International Conference on Image Processing, Washington, DC, USA, 23-26 Oct. 1995.

[48] R.R. Schultz, R.L. Stevenson, "Improved Definition Image Expansion", Proc. of IEEE Acoustic, Speech, and Signal Processing Conference, San Francisco, CA, USA, March, 23-26, 1992.

[49] T. Sikora, P.Richardson, "Image Interpolation Based on Image Statistics: a Case Study", Proc. of IEEE Region 10 International Conference, TENCON '92, Melbourne, Vic., Australia, Nov. 11-13, 1992.

[50] S.A. Martucci, "Image Resizing in the Discrete Cosine Transform Domain", Proc. of International Conference on Image Processing, Washington, DC, USA, Oct. 23-26, 1995.

[51] N. Merhav, V. Bhaskaran, "A Transform Domain Approach to Spatial Domain Image Scaling", Proc. of IEEE Acoustic, Speech, and Signal Processing Conference, Atlanta, GA, USA, May 7-10, 1996.

[52] K.P. Hong; J.K. Paik; H.J. Kim; C.H. Lee, "An Edge-Preserving Image Interpolation System for a Digital Camcorder", IEEE Trans. on Consumer Electronics, Vol. 42, No. 3, Aug. 1996, pp. 279-284.

[53] K.P. Hong; J.H. Paik; H.J. Kim; C.H. Lee, "An Edge-Preserving Image Interpolation System for a Digital Camcorder", Proc. of IEEE International Conference on Consumer Electronics, Rosemont, IL, USA, June 5-7, 1996.

[54] F. Ahmed; S.C. Gustafson; M.A. Karim, "High-Fidelity Image Interpolation Using Radial Basis Function Neural Networks", Proc. of IEEE National conference on National Aerospace Electronics, Dayton, OH, USA, May 22-26, 1995.

[55] T. Sigitani, Y. Iiguni, H. Maeda, "Image Interpolation for Progressive Transmission by Using Radial Basis Function Network", IEEE Trans. on Neural Networks, Vol. 10, No. 2, Marsh 1999, pp. 381–390.

[56] E. Gelenbe, H. Bakircioglu, T. Kocak, "Iamge Processing with the Random Neural Network (RNN)", Proc. of 13th International Conference on Digital Signal Processing, July 2–4, 1997.

[57] S.G. Chang, Z. Cvetkovic, M. Vetterli, "Resolution Enhancement of Images Using Wavelet Transform Extrema Extrapolation", Proc. of IEEE International Conference on Acoustics, Speech, and Signal Processing, ICASSP-95, New York, NY, USA, May 9-12, 1995.

[58] W.K. Carey; D.B. Chuang; S.S, Hemami, " Regularity-Preserving Image Interpolation", Proc. of International Conference on Image Processing, Oct. 26-29, 1997.

[59] A. Youssef, "Analysis and Comparison of Various Image Downsampling and Upsampling Methods", Proc. of IEEE Conference on Data Compression, Snowbird, UT, USA, March 30-April 1, 1998.

[60] Kreyszic, "Advanced Engineering Mathematics", sixth edition, John Wiley & Sons, 1988.

[61] M. Abramowitz; I.A. Stegun, "Handbook of Mathematical Functions", Dover Edition, General Puplishing Company, Ltd, Toronto, Ontario, Canada, 1965

[62] Alen W. Paeth, "Graphics Gems V", Academic Press Professional Inc., 1995.

[63] Foley; Van Dam; Feiner; Hughes; Phillips, "Introduction to Computer Graphics", Addison-Wesley Publishing Company, Inc., 1994.

[64] F.S Hill Jr., "Computer Graphics", Macmillan Publishing Company, 1990.

[65] R.C. Gonzalez; P. Wintz, "Digital Image Processing", Addison-Wesley, 1987.

[66] J. Canny, "A Computational Approach to Edge Detection", IEEE Trans. on Pattern Anal. and Mach. Intelligence, vol. 8, pp 679-698, 1986.

[67] D. Marr; H. Hildreth (1980), "Theory of Edge Detection", Proc. Roy. Sco. London, B207, 187-217.

[68] S. Ligin; S. Dinaggang; Q. Feih, "Edge Detection on Real Time Using LoG Filter", IEEE Proc. of International Symposium on Speech, Image Processing, and Neural Network, Shanghai, China, 13-16 April, 1994.

[69] A. A. Masoud; M. M. Bayoumi, "Using local structure for the reliable removal of noise from the output of the LoG edge detector", IEEE Trans. Systems, Man and Cybernetics, Vol. 25, No. 2, Feb 1995, pp. 328–337.

[70] L. Ganesan; P. Bhattaharyya, "Edge Detection in Untextured and Textured Images – a Common Computational Framework", IEEE Trans. on Systems, Man and Cybernetics, Part B, vol. 27, pp 823-834, 1997.

[71] N. Okamoto; W. Chen; N. Iida; T. Minami, "Automatic extraction of contour lines and feature points from profile images", Proc. of IEEE Canadian Conference on Electrical and Computer Engineering, May 25–28, 1997.

[72] Y. Haifeng, H. Makela, "A new edge extraction method for vehicle navigation", Proc. of ICAR Fifth International Conference on Advanced Robotics, June 19–22, 1991.

[73] F. Mokhtarain, R. Suomela "Curvature scale space for robust image corner detection", Proc. of 14th International Conference on Pattern Recognition, Aug 16–20, 1998.

[74] L. Bin, A. D. J. Cross, E. R. Hancock, "Corner detection using vector potential", Proc. of 14th International Conference on Pattern Recognition, Aug 16–20, 1998.

[75] F. Mokhtarian, and R. Suomela, "Robust Image Corner Detection Through Curvature Scale Space", IEEE Trans. on Pattern Analysis and Machine Intelligence, vol. 20, pp 1376-1381, 1998.

[76] P.E. Trahanias, A.N. Venetsanopoulos, "Color Edge Dtection Using Vector Order Statistics", IEEE Trans. Image Processing, Vol. 2, No. 2, April 1993, pp. 259-264.

[77] P. S. Wu, L. Ming, "Fast edge detection for color image", Proc. of 3rd International Conference on Signal Processing, Oct 14–18, 1996.

[78] G. C. Gurski, M. T. Orchard, "On the use of orientation information for improved contour difference coding", Proc. of IEEE Symposium on Circuits and Systems, May 3–6, 1993.

[79] J. Yuan, C.Y. Suen, "An Optimal Algorithm for Detecting Straight Lines in Chain Code", IAPR international Conference on Pattern Recognition. Montreal, Que, CANADA, 30 Aug. – 3 Sept., 1992.

[80] A.K. Murad Agha; R.K. Ward; S. Zahir, "Image Expansion Using Segmentation-Based Method", Proc. of IEEE Pacific Rim Conference on Communications, Computers and Signal Processing, PACRIM 1999, Victoria, Canada, August 22-24, 1999.

[81] A.K. Murad Agha; R.K. Ward; S. Zahir, "A Near Exact Image Expansion Scheme for Bi-Level Images", Proc. of International Conference on Image Processing, ICIP'2000, Vancouver, Canada, September 10-13, 2000.

[82] J. Koplowitz, and V. Greco, "On the Edge Location Error for Local Maximum and Zero-Crossing Edge Detector", IEEE Trans. on Pattern Analysis and Machine Intelligence, vol. 16, pp 1207-1212, 1994.

[83] H. Li, B.S. Manjunath, S.K. Mitra, "Contour – Based Multisensor Image Registration", IEEE Proc. of the Twenty Sixth Asilomar Conference on Signals, Systems, and Computers. Santa Barbara, CA, USA, 26-28 Oct.. 1992.

[84] M. Gokmen, C. C. Li, "Edge detection and surface reconstruction using refined regularization", IEEE Trans. Pattern Analysis and Machine Intelligence, Vol. 15, No. 5, May 1993, pp. 492–499.

[85] D. E. Tamir, K. Phillips, A. Abdul-Karim, "Efficient chain-code encoding for segmentation-based image compression", Proc. of DCC Conference on Data Compression, Marsh 31–April 3, 1996.

[86] "A Study Guide for Digital Image Processing", Mark J.T. Smith, Alen Doecf, Scientific Publishers, 1997

[87] "Digital Image Processing Techniques", Michael P. ekstrom, Academic Press, 1984.

[88] "Fundamentals of Digital Image Processing", Anil K. Jain, Prentice Hall, 1989.

[89] "Numerical Recipes Example Book (C)", W.T. Vetterling; S.A. Teukolsky; W.H. Press; B.P. Flannery, Cambridge University Press, 1988.