IMAGE INTERPOLATION WITH HIGH VISUAL QUALITY

by

QING WANG

M. Sc., Xidian University, 1994

B. Sc., Xidian University, 1990

A THESIS SUBMITTED IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE DEGREE OF

DOCTOR OF PHILOSOPHY

In

THE FACULTY OF GRADUATE STUDIES
Department of Electrical and Computer Engineering

THE UNIVERSITY OF BRITISH COLUMBIA

December 2004

Abstract

Image interpolation or resizing has wide applications in areas such as digital photography, the printing industry, HDTV and computer graphics. The traditional methods for image interpolation are known to suffer from visual artifacts that degrade the subjective quality of images. The goal of this research is to study and develop new image interpolation techniques that provide images with improved visual quality. In our study, we focus on the two most harmful artifacts resulting from traditional image interpolation, zigzagging and blurring.

In developing the new interpolation method, we concentrate on visually oriented interpolation techniques. We explore the feature-oriented and content-adaptive approach emerging from recent studies on image interpolation.

To find out why and how zigzagging arises, we first study the isophotes, i.e., the equi-intensity contours, of interpolated images. Based on this analysis, we design an interpolation scheme that employs interpolation grid that adapt to the orientation of image isophotes. This method yields much smoother isophotes in the interpolated images than the traditional interpolation methods. As a result, the zigzagging artifacts are largely suppressed.

To remove the blurring effect, we propose a method of enhancing the contrast of expanded images. To do so, we first discuss the properties an edge-sharpening function should satisfy. From these we choose a family of edge-sharpening functions that have the desired properties. The proposed contrast enhancement method proves to be effective in removing the blur.

We also show that evaluations based on the mean squared error (MSE) are not effective, thus we employ and improve the curvature-based measures to evaluate the performance of the proposed interpolation method. The experiments and analysis show that our proposed interpolation method is visually superior to traditional methods, and provide interpolated images with high visual quality.

Table of Contents

List of Tables

List of Figures

Glossary

Symbol	Meaning
a	Usually a free parameter of a function.
$c(x, y)$	The curvature of isophote at point (x, y).
$f(x, y)$	The original continuous image.
f_x, f_{xx}	First and second derivatives of function $f(x, y)$.
$\hat{f}(x, y)$	The reconstructed continuous image, an approximation of $f(x, y)$.
$F_d(m, n)$	The original digital image, sampled from the original continuous image.
$F(m, n)$	The digital image acquired from $F_d(m, n)$ through interpolation.
$\vec{G}(x, y)$	Gradient of a 2-variable function at point (x, y).
$h(x, y)$	The interpolation function.
I_0, z_0	Refer to given intensity values.
R	The interpolation factor. For enlargement $R>1$; for reduction $R<1$.
Δ_x, Δ_y	The horizontal and vertical sampling intervals.
$\varphi_a(x)$	The edge-sharpening function with parameter a.
θ	An angle, usually that of the interpolation parallelogram.
σ	Standard deviation.
∇	The gradient operator.
$\| \|$	The norm operator.

Acknowledgements

I am most grateful to my supervisory committee for their guidance, advice and support throughout this work. In particular, I am genuinely grateful to Dr. Rabab Ward, my supervisor, for the patient guidance, resourceful discussions and sound advices I received from her. This research work, and the author himself, have greatly benefited from her academic excellence and strictness, and the academic freedom she provided to this research.

I am also thankful to Dr. Ian Cumming and Dr. Mathew Yadlin, for the advice they provided in completing the proposal of this research. In addition, I am appreciative to Dr. Nawaf Karhma for the discussions that broadened my scope of view.

I would like to thank Dr. Hongjian Shi and Huiqun Deng, who provided helpful discussions and advices in many technical issues, including programming, experimenting and mathematics. I also would thank Dr. Jiancheng Zou for his help in mathematics.

Lastly, I would like to thank Dr. Mehran Azimi, Mehrdad Fatourechi, and other staff and members of the Image Lab and the Department of ECE, who have assisted me during this research work.

Chapter 1. Introduction

1.1 Background

In many image processing applications, it is common to seek a digital image with a higher spatial resolution than that of a given image. During the process of changing the spatial resolution of a digital image, intensity values of newly introduced pixels (known as ***children pixels***) have to be estimated from those of the original grid pixels (known as ***parent pixels***). The operation of estimating children pixels from the existing parent pixels is called ***image interpolation***. In our research, we are only interested in image interpolation using digital techniques, instead of analog or optical methods.

When referring to image interpolation different researchers use different terms, such as image resizing, re-sampling, zooming, super-resolution and sub-resolution, up-sampling and down-sampling, expansion (or enlargement, magnification) and reduction. Usually these terms are used to emphasize different aspects of image interpolation, depending on the intended application.

In fact, image interpolation is not restricted to changing the spatial resolution. Another application of image interpolation is image rotation. When an image is rotated, a point (s,u) in the rotated lattice is related to point (x, y) in the original lattice by

$$\begin{bmatrix} s \\ u \end{bmatrix} = \begin{bmatrix} \cos\theta & \sin\theta \\ -\sin\theta & \cos\theta \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix}$$

For an arbitrary angle θ, pixels in the rotated coordinates usually are not in grid positions in the original coordinates, therefore their values are not known and must be estimated. The pixel values in the rotated image are obtained through image interpolation.

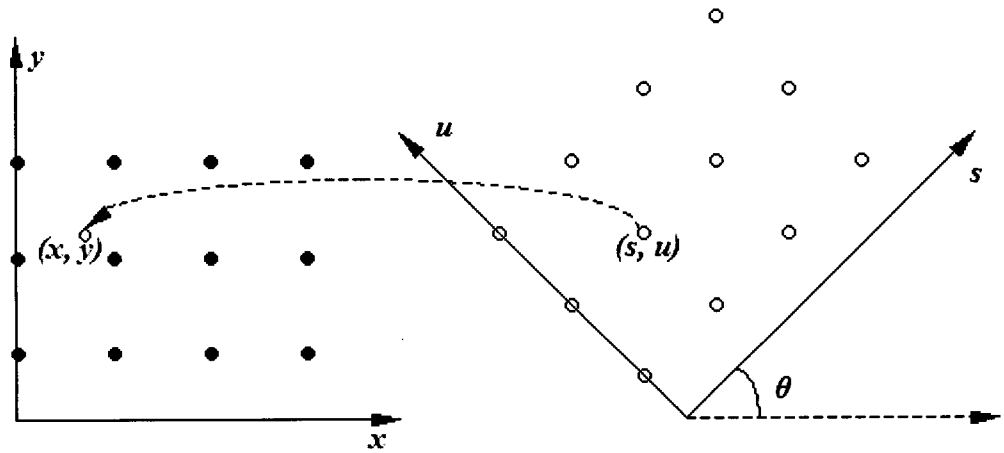


Figure 1.1 Rotation of coordinates

Point *(s, u)* in the rotated grid (right) corresponds to point *(x, y)* in the original grid, the value of which is unknown and has to be interpolated.

Image interpolation has applications in many areas, including

- **Resizing** and **rotation** of digital images. These are vital operations in digital camera, image workshops and digital photography.
- **Large format printing**. In the advertisement industry, production of the large-scale prints is currently a very expensive, computationally demanding and time-consuming process. One important reason is the lack of efficient image interpolation techniques.
- **Resolution conversion** amongst different TV systems, such as the present TV broadcasting and the HDTV. The compatibility between low-resolution transmitters and high-resolution receivers and vice versa also require conversion of resolution, thus video interpolation is needed.

- **Computer graphics**. In 3-D graphical applications (e.g., in entertainment and computer game industries), rendering of 3-D objects requires color and shadowing information to be obtained from the known discrete pixels, a special type of interpolation referred to as triangulation. Currently, effective and visually pleasing triangulation remains a highly researched topic in the field of computer graphics.
- **Video compression/Zooming**. By deriving more pixels from a limited set of known pixels, image interpolation is useful in compressing images. An image's size may be reduced before compressing it. The reconstructed image must then be enlarged to its original size. Interpolation-based compression is especially useful when combined with multi-scale video transmission.

1.2 Objectives

In the past few decades, substantial research effort has been made in the study of effective interpolation methods. However, the majority of the current interpolation methods are still based on the traditional techniques. As we will discuss in the next two chapters, traditional interpolation methods suffer from visual degradations, or artifacts. As a result, the images interpolated through traditional methods often have inferior visual qualities.

In recent years, research interest in image interpolation has been more focused on visually oriented techniques, that is, the new emphasis is to develop interpolation

methods that provide images with better visual quality than the traditional methods. In the recently introduced interpolation methods, different techniques have been proposed to study and solve the visual degradations associated with traditional interpolation, of which the zigzagging artifacts and the blurring effects are of most interest.

In this thesis, we propose a new visually oriented interpolation method that provides images with superior visual quality. To achieve our goal, we carry the following studies:

1. Survey the recently introduced interpolation methods, especially the visually oriented methods.
2. Study the zigzagging artifacts by examining the isophotes (equi-intensity contours) of the interpolated images.
3. Develop an isophote-oriented, orientation-adaptive interpolation scheme that provides smooth isophotes in the interpolated images. By doing so, the zigzagging artifacts are largely reduced.
4. Propose a content-adaptive scheme of image sharpening, which enhances the contrast of the interpolated images, thus removing the blurring effects.
5. Evaluate the performance of the proposed interpolation method by conducting experiments and quantitative analysis based on improved curvature measures.

1.3 Structure of the Thesis

In Chapter 2, we review the traditional study of interpolation. We summarize the traditional interpolation kernels and the associated interpolation methods. Images

interpolated through the traditional methods are shown. We also discuss the traditional MSE-based evaluation of image quality in this chapter.

In Chapter 3 we first discuss the degradations associated with the traditional methods — the zigzagging and blurring artifacts. Next, the recently introduced interpolation methods are surveyed. Most of these methods emphasize the visual quality of images, thus we classify as "visually oriented" methods. We discuss the general ideas as well as the specific approaches of these methods, and then analyze their strengths and weaknesses.

In Chapter 4, we examine the isophotes of interpolated images in order to explain and solve the zigzagging artifacts. We introduce a method for detecting the isophotes of the bi-linear and bi-cubic interpolated images. Of special interest, we study the curvature of the isophotes in the bi-linearly interpolated images.

In Chapter 5, we propose a directional interpolation method, which is based on the isophote analysis in Chapter 4. In this method, the bi-linear interpolation is generalized by employing interpolation lattice that adapts to the orientation of isophotes. We show that by doing so the interpolated isophotes are smoother than those of the traditional bi-linear method, thus suppressing the zigzagging artifacts. The implementation of the proposed interpolation method is described, and experimental results are shown. Also in Chapter 5, we conduct a curvature evaluation of different interpolation methods, showing the superiority of the proposed method.

Chapter 6 introduces a scheme of contrast enhancement for removing the blurring effect in the interpolated images. The proposed scheme employs different sharpening

techniques in edge areas and smooth areas. The algorithm of the proposed sharpening method is described and experimental results are shown.

In Chapter 7, we summarize and discuss the proposed interpolation method. We summarize the contributions of this thesis, and give suggestions for future research.

Chapter 2.
TRADITIONAL INTERPOLATION METHODS

In this chapter, we review the traditional interpolation methods. Section 1 discusses the image interpolation as an image re-sampling problem, and briefly reviews the traditional study of image interpolation. Section 2 and Section 3 review the traditional spatial domain methods and transform domain methods, respectively. Section 4 shows the experimental results obtained through some of the traditional methods reviewed. In section 5, we discuss the MSE measurement of interpolation methods.

2.1 Image Interpolation as a Re-sampling Problem

2.1.1 The Sampling Theorem

In the study of image interpolation, the digital image to be expanded or reduced, $F_d(m, n)$, is usually assumed to be acquired by sampling an original continuous image $f(x, y)$. Ideally, the resulting interpolated image $F(m, n)$ should be obtained by re-sampling $f(x, y)$ at the new spatial resolution or on the new spatial lattice. Usually, the original continuous image $f(x, y)$ is unknown, thus it has to be estimated from its known samples $F_d(m, n)$.

According to the Nyquist sampling theorem, if an image $f(x, y)$ is band-limited, i.e.,

$$F(\varpi_1, \varpi_2) = 0, \quad \text{for} \,|\varpi_1| > \Omega_1 \text{ or} \,|\varpi_2| > \Omega_2 \tag{2-1}$$

and the sampling intervals Δ_x and Δ_y satisfy

$$\frac{1}{\Delta_x} > 2\Omega_1 , \frac{1}{\Delta_y} > 2\Omega_2 , \tag{2-2}$$

then the original image *f(x,y)* can be precisely reconstructed from its samples by

$$f(x,y) = \sum_{m=-\infty}^{\infty} \sum_{n=-\infty}^{\infty} F_d(m,n) \cdot \mathrm{sinc}(x-m)\mathrm{sinc}(y-n) \tag{2-3}$$

In (2-3) the *sinc* function is defined as

$$\mathrm{sinc}(x) = \frac{\sin(\pi x)}{\pi x}$$

For convenience and without losing generality, we assume that the sampling intervals Δ_x and Δ_y for the original image are always 1 unless otherwise specified.

Figure 2.1 and 2.2 show the waveforms of the one-dimensional and two-dimensional *sinc* functions, respectively. In (2-2), $2\Omega_1$ and $2\Omega_2$ are called the Nyquist sampling rates.

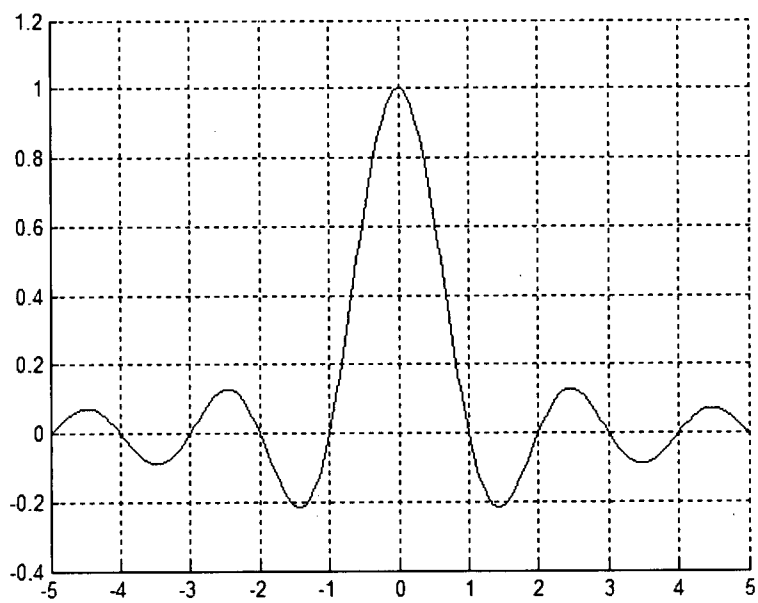


Figure 2.1 The one-dimensional *sinc* function

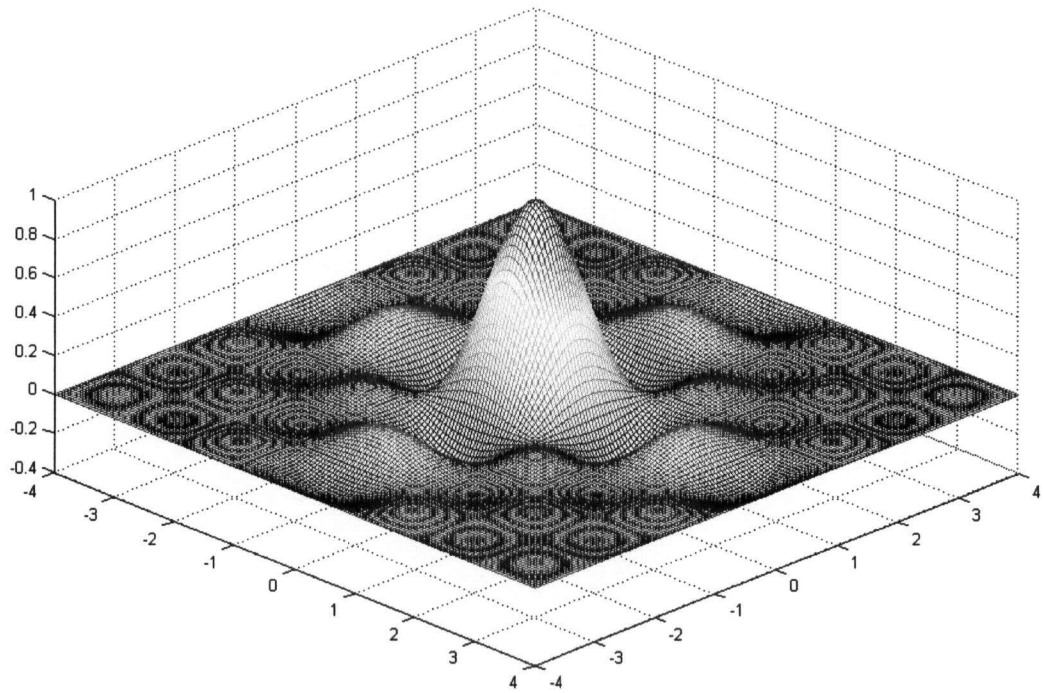


Figure 2.2 The two-dimensional *sinc* function

When reconstructing the original image (formula (2-3)), the sampling positions for the original continuous image, i.e., the original sampling grid, must be known. In practice, for a given original digital image $F_d(m,n)$, the exact layout of the original sampling grid is often not known, and usually has to be assumed. In this thesis, we make the following general assumptions on the sampling grid.

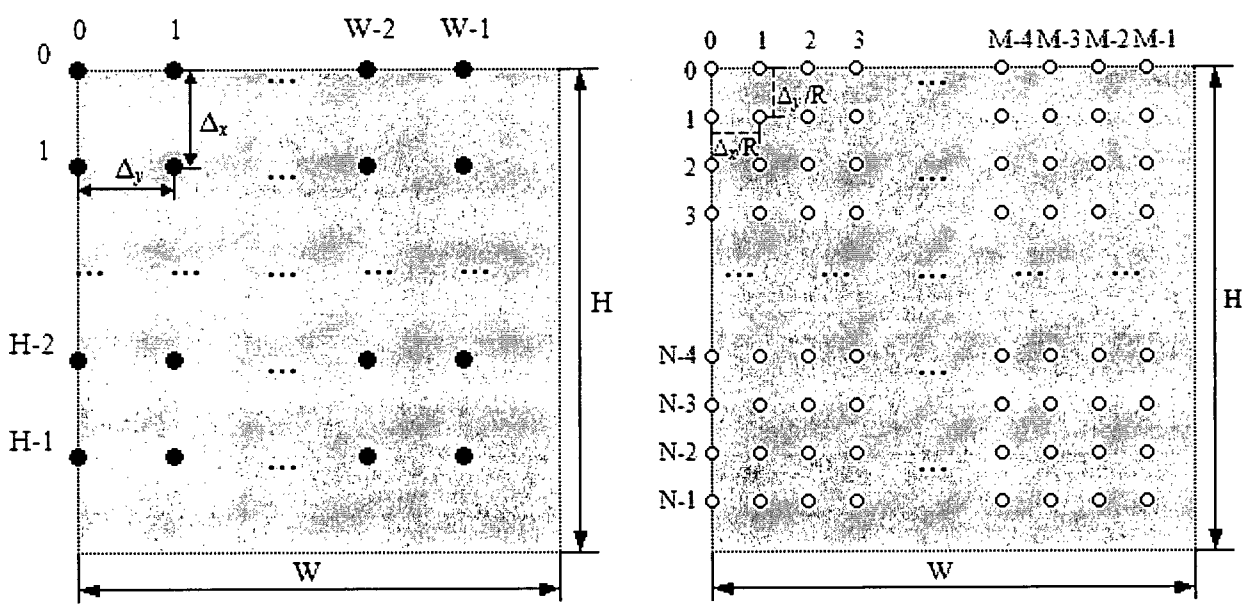
(1) The sampling grid is square-based, that is, the samples are placed in positions with coordinates ($m\Delta_x$, $n\Delta_y$), where m and n are non-negative integers. We also use integer pair (m, n) as the index of a sampling position or a pixel.

(2) The pixel with index (0, 0), i.e., the top-left sampling position, is always positioned on the point in the continuous image that has coordinates (0, 0), i.e., the top-left corner of the continuous image.

When re-sampling the reconstructed continuous image, a new sampling grid needs to be set. When forming the new sampling grid, we also follow the same assumptions as above, except that the spacing between sampling positions are different from the original image. For expansion or reduction of an image with an interpolation ratio of R>0, the new horizontal and vertical spacing between pixels are Δ_x/R and Δ_y/R, respectively. The sampling grids are illustrated in Figure 2.3.

In Figure 2.3, the shaded area is the continuous image. The original continuous image is assumed to have the size of W (width) × H (height). Assuming $\Delta_x=1$ and $\Delta_y=1$, the original digital image contains exactly W×H pixels. See Figure 2.3(a).

In figure (b), the reconstructed continuous image has the same size as the original continuous image, but the re-sampling grid is different from the original sampling grid. The new sampling grid contains M×N pixels, where M=W×R and N=H×R (after rounding to the nearest integers if necessary).



(a) Sampling grid for the original image (b) Sampling grid for interpolated image

Figure 2.3 Illustration of sampling grid

2.1.2 The Practical Interpolation Functions

The Nyquist sampling theorem indicates that a band-limited original image $f(x, y)$ can be precisely reconstructed from its samples $F_d(m, n)$ if the sampling rate is no lower than the Nyquist rate. The interpolated image $F(m, n)$ can in turn be "ideally" obtained by re-sampling the reconstructed $f(x, y)$. Unfortunately, due to well-known reasons, the sinc function required by the Nyquist reconstruction (equation (2-3)) is impossible to implement. Therefore, in practice, different functions have been developed to replace the sinc function in (2-3), yielding

$$\hat{f}(x,y) = \sum_{m=-\infty}^{\infty} \sum_{n=-\infty}^{\infty} F_d(m,n) \cdot h(x-m, y-n) \qquad (2\text{-}4)$$

In (2-4), $\hat{f}(x,y)$ is the approximately reconstructed continuous image, an approximation of $f(x, y)$. Function $h(x, y)$ is called the interpolation function. In our study, we also refer to $h(x, y)$ as the interpolation kernel.

When developing a practical interpolation function $h(x,y)$, usually $h(x, y)$ is required to meet certain restrictions so that it has desirable properties for the interpolation purpose. In our research, we survey the restrictions adopted by different researchers, and summarize some of the commonly used ones as follows.

An interpolation function $h(x,y)$ is or has

1. Similar waveform as the sinc function. $h(x, y)$ can also be developed in the Fourier domain. In this case, the spectrum of $h(x, y)$ is required to resemble the rectangular impulse, the spectrum of the sinc function. The resemblance is often measured by the percentage of energy contained in the main lobe, or the speed of attenuation of the side lobes.

2. Compactly supported. This is a critical condition for *h(x, y)* to be practical. Most interpolation functions should have zero value beyond a small region of support. Some functions, which are originally defined to have large or infinite region of support, are truncated so as to become compactly supported.

3. Separability. *h(x,y)* is separable if

$$h\ (x, y) = h_1(x) \cdot h_2(y) \qquad (2\text{-}5)$$

Separable functions are often mathematically easier to handle than inseparable ones. More importantly, separability of interpolation functions leads to the separability of the interpolation process, that is, interpolation could be done first in one dimension and then the other. This property usually simplifies the interpolation algorithm.

4. C_i continuous. A function is C_i continuous if its i-th derivative is continuous. Usually *h(x, y)* has different requirements of continuity in different applications. For example, if the curvature of $\hat{f}(x, y)$ is to be analyzed, then it is highly desirable that the interpolation function *h(x,y)* is C_2 continuous.

5. Straightness, that is, for integers *m* and *n*,

$$h(m,n) = \begin{cases} 1, & m = n = 0 \\ 0, & \text{otherwise} \end{cases} \qquad (2\text{-}6)$$

To meet the straightness restriction, *h(x, y)* must have zero values on grid positions except point (0, 0). The straightness property guarantees that the values of the sampled pixels remain unchanged after interpolation.

6. Energy conservation. That is,

$$\iint h(x, y)dxdy = 1 \qquad (2\text{-}7)$$

If $h(x,y)$ meets the energy conservation condition, the average intensity level of the image is constant before and after interpolation.

Note that the above restrictions are neither necessary nor sufficient conditions for function $h(x,y)$ to be a practical interpolation kernel. For many traditional kernels, the above restrictions are not all met at the same time. In fact, usually some of the restrictions have to be sacrificed for others to be met. For example, in the piecewise polynomial kernels, usually higher differentiability (restriction 4) results in a greater region of support, and therefore restriction 2 (compactly supported) is sacrificed.

One property that all the traditional kernels reviewed in this chapter have in common is separability, that is, all these two-dimensional kernels have the form of

$$h\ (x, y) = h_1(x) \cdot h_1(y)$$

It is also common that extra restrictions are supplemented to those listed above, in order to achieve other desired properties under the given circumstance. For example, $h(x,y)$ may be required to have linear phase in the Fourier domain, in order to eliminate the phase distortion in interpolation.

Each different $h(x, y)$ yields a different reconstructed image $\hat{f}(x, y)$ through equation (2-4). That is to say, each interpolation kernel corresponds to a unique interpolation method. In the next section, we review some of the commonly used interpolation methods. Many of these methods were developed in the early stage of the study of image interpolation, thus are known as the traditional interpolation methods. In our review, the traditional methods are divided into two main categories, the spatial domain methods and the transform domain methods.

2.2 Spatial Domain Interpolation Methods

Among the traditional spatial domain methods, the most commonly used are:

(1) The piecewise polynomials,

(2) The B-Spline functions, and

(3) The Gaussian functions.

In the following sections, we review each of the above methods.

2.2.1 Piecewise Polynomial Kernels

Piecewise polynomial kernels form the largest family of traditional interpolation functions, and have received extensive applications in image interpolation. The polynomial kernels are identified by their degrees. Limited by computational complexity, usually polynomials with 0 to 3rd degrees are used in practice, while those with higher degrees are rarely used. The interpolation method associated with piecewise polynomial kernels are commonly referred to as pixel replication (degree 0), bi-linear interpolation (degree 1), quadratic interpolation (degree 2), and bi-cubic interpolation (degree 3). The polynomial kernels are reviewed below.

Pixel replication

The simplest piecewise polynomials kernel is the rectangular impulse function, that is

$$h^{rep}(x) = \begin{cases} 1 & -\frac{1}{2} < x \le \frac{1}{2} \\ 0 & otherwise \end{cases} \tag{2-8}$$

In the 0 order interpolation, the values of the children pixels (pixels to be interpolated) are obtained by replicating the nearest parent pixel. This method is also referred to as the "nearest neighbor" method.

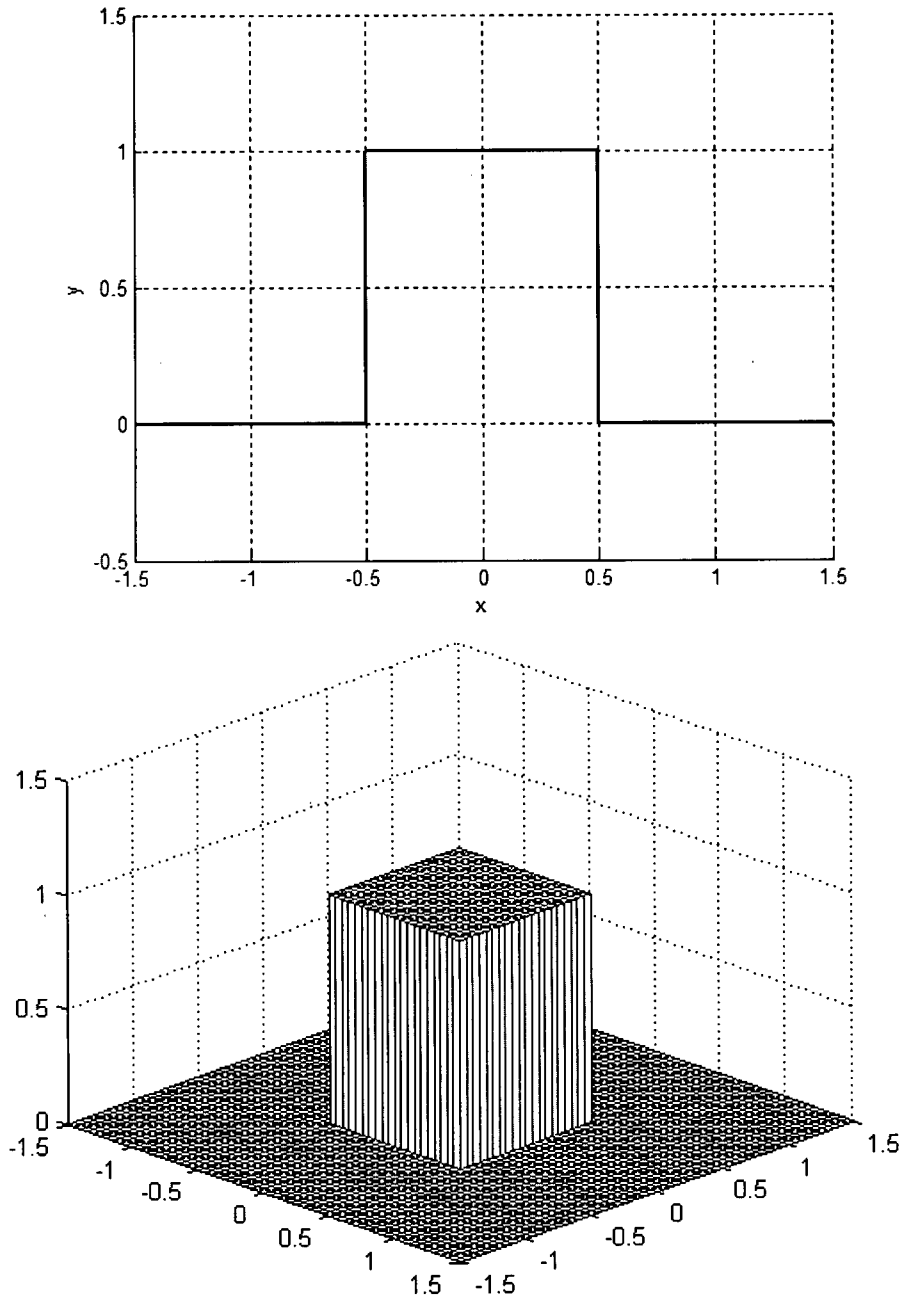


Figure 2.4 Kernel of pixel replication.

Bi-linear interpolation

The kernel of bi-linear interpolation is the triangle impulse, that is

$$h^{linear}(x) = \begin{cases} 1-|x| & |x| \le 1 \\ 0 & otherwise \end{cases}. \qquad (2\text{-}9)$$

The 1-dimensional and 2-dimensional forms of the bi-linear kernel are shown in Figure 2.5.

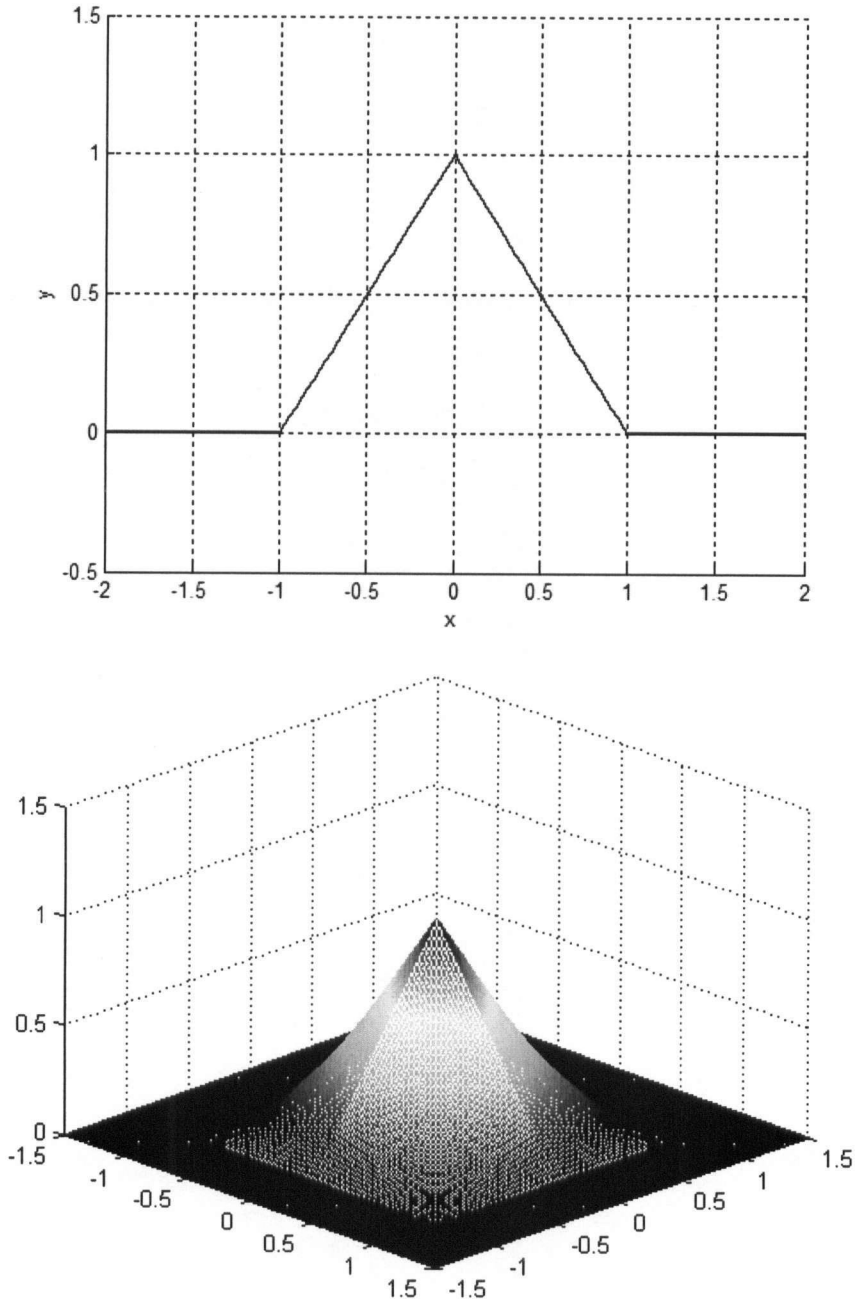


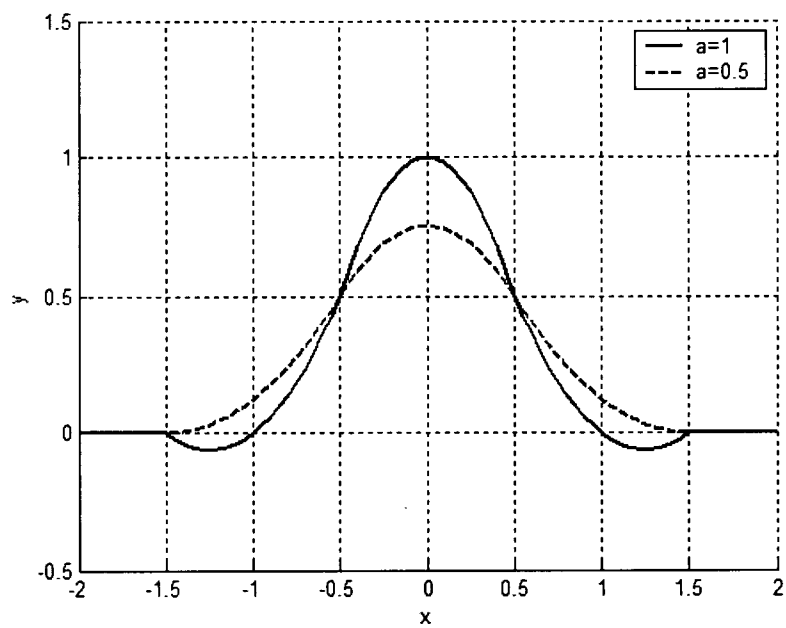
Figure 2.5 Kernel of bi-linear interpolation.

Quadratic interpolation

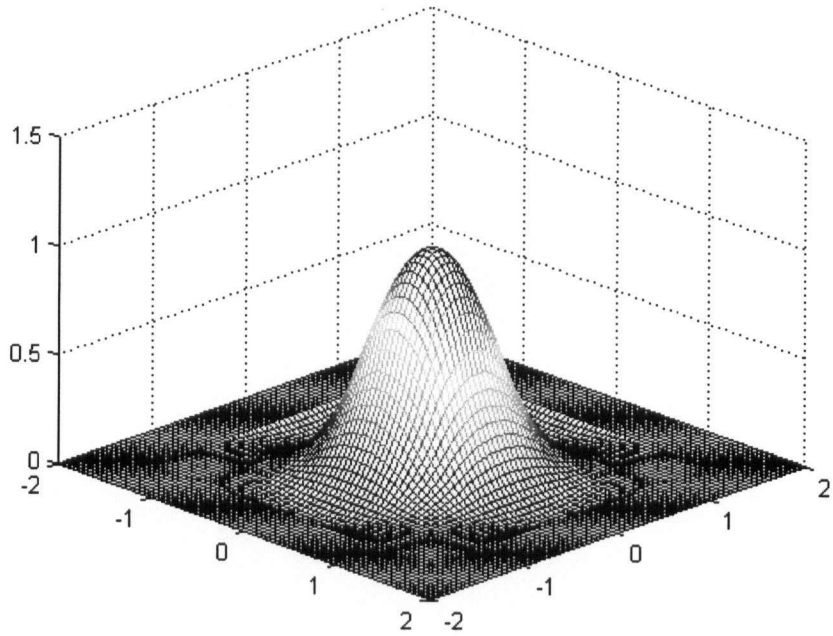
The quadratic interpolation kernel proposed in [32] is

$$h^{quad}(x) = \begin{cases} -2a|x|^2 + \frac{1}{2}(a+1) & |x| < \frac{1}{2} \\ a|x|^2 + (-2a - \frac{1}{2})|x| + \frac{3}{4}(a+1) & \frac{1}{2} \le |x| < \frac{3}{2} \\ 0 & \text{otherwise} \end{cases} \qquad (2\text{-}10)$$

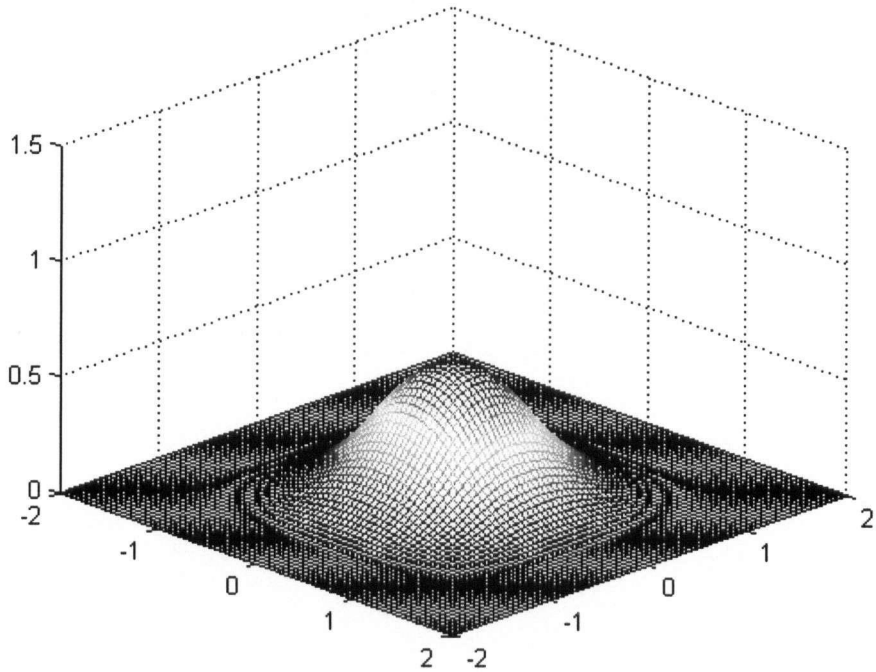
In (2-10) a is a free parameter that gives the interpolation function a one-degree freedom. Regardless of the value of a, restriction 6 (energy conservation) is met, whereas restriction 4 (C_1 continuous) and 5 (Straightness) cannot be both met at the same time. With $a=1$ restriction 5 is met while 4 is not; with $a=0.5$ restriction 4 is met while 5 is not. In practice, a is usually chosen to be either 1.0 or 0.5 depending on the specific application. Figure 2.6 shows the kernels of the quadratic interpolation.



(a) The 1-d quadratic kernels



(b) 2-d kernel with *a=1*.



(c) 2-d kernel with *a=0.5*.

Figure 2.6 Kernel of quadratic interpolation

Note that when $a=1.0$, for children pixels with coordinates ($x=n$ or $n+0.5$, $y=m$ or $m+0.5$), where n and m are integers, quadratic interpolation yields the same results as the bi-linear interpolation. This implies that when expanding an image by 2.0×2.0, the quadratic interpolation with $a=1.0$ is equal to bi-linear interpolation.

Compared with other polynomial-based methods, quadratic interpolation has received less attention and research interest. One reason for this is the phase distortion introduced by the quadratic kernel ([8]). However, some researchers ([32]) showed that the properly designed quadratic interpolation can provide acceptable interpolation results with low to moderate computational complexity.

Bi-cubic interpolation

Cubic interpolation kernels have received the most research attention among all polynomial kernels. In the past few decades, many types of different cubic kernels have been studied and introduced.

The most commonly used 1-dimensional cubic interpolation function is Keys' cubic spline ([10]), where,

$$h_1^{Keys}(x)=\begin{cases}(a+2)|x|^3-(a+3)|x|^2+1 & 0\le|x|\le1\\ a|x|^3-5a|x|^2+8a|x|-4a & 1\le|x|\le2\\ 0 & |x|>2\end{cases} \qquad (2\text{-}11)$$

$h_1^{Keys}(x)$ is a four-point interpolation kernel, which means it uses the value of the four nearest grid pixels to interpolate a child pixel.

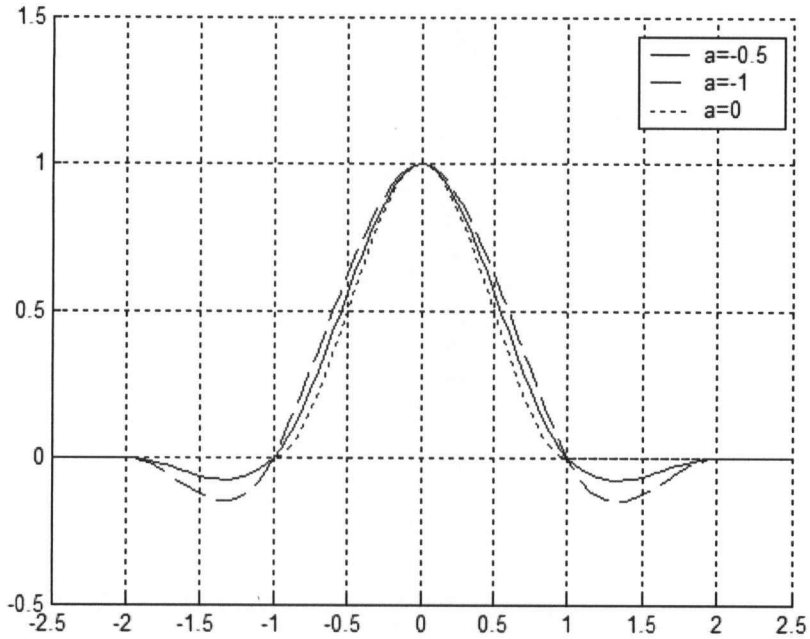
Keys' cubic spline has a free parameter a, which controls the shape and properties of the cubic spline. Regardless of the value of a, $h_1^{Keys}(x)$ is C_1 continuous and has the

energy conservation and straightness properties, i.e., restriction 4, 5 and 6 are all unconditionally met. Therefore, the value of parameter a is decided by another restriction.

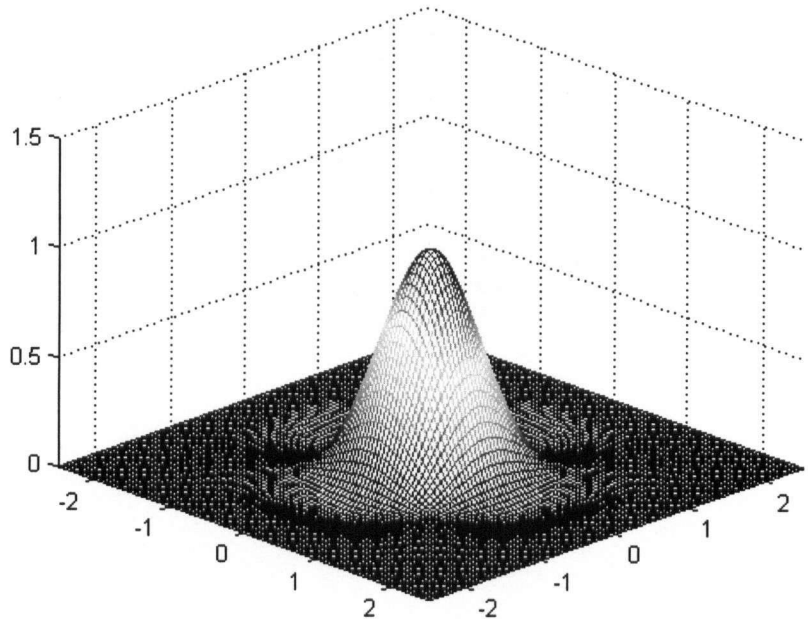
Keys ([10]) proved that with a=-0.5, the coefficients of the Taylor expansion of $h_1^{Keys}(x)$ are the same as the *sinc* function up to the third (x^2) term. This property implies that the Keys' cubic with a=-0.5, known as Catmull-Rom cubic spline, is able to precisely interpolate (reconstruct) any original function $f(x,y)$ which does not have terms higher than x^2 in its Taylor expansion. Catmull-Rom cubic spline is one of the most frequently used cubic splines in image interpolation and computer graphics.

$h_1^{Keys}(x)$ with other values of a are also studied in order to meet specific requirements useful for certain applications. For example, when a=1 $h_1^{Keys}(x)$ has the same slope with the *sinc* function at x=1, whereas a=-3/4 makes the second derivative of $h_1^{Keys}(x)$ continuous at x=1. Regardless of the value of a, Keys' cubic spline is NOT C_2 continuous.

Other than the family of Keys' cubic spline, 6-point and 8-point cubic kernels, $h_1^{cubic6}(x)$ and $h_1^{cubic8}(x)$, are also studied ([35], [36]). These cubic kernels involve the nearest 6 and 8 parent pixels, respectively, to interpolate a child pixel. With a larger region of support and more degrees of freedom, these cubic kernels are able to meet stricter restrictions, such as C_2 continuity, than the Keys' cubic. However, due to their high computational cost, the 6-point and 8-point kernels are not widely used in practice.



(a) 1-d Keys' cubic functions



(b) 2-d Catmull-Rom kernel.

Figure 2.7 The Keys' cubic kernel

Due to their high complexity, piecewise polynomial kernels with degrees higher than 3 are rarely attempted by researchers. As a result, these interpolation kernels are not reviewed.

2.2.2 The B-Spline Kernels

Basic splines (B-splines) are amongst the most commonly used in the family of spline functions. The B-splines can be constructed by a series of self-convolutions of a basis function, that is,

$$h_{Bspline}^{N}(x) = \mu(x) \underbrace{* \mu(x) * \ldots * \mu(x)}_{N-1\ times} \tag{2-12}$$

In (2-12) $h_{Bspline}^{N}(x)$ is the Nth-order B-spline function, and $\mu(x)$ is the basis function. Usually, simple and impulse-like functions are chosen as the basis functions.

If using the kernel of the pixel replication $h_1^{rep}(x)$ (the rectangular impulse) as the basis function, the linear kernel $h_1^{linear}(x)$ in (2-9) is the 2nd order (N=2) B-spline function, and the quadratic kernel $h_1^{quadr}(x)$ in (2-10) with a=0.5 is the 3rd order B-spline. The cubic (N=4) B-spline is

$$h_{BSpline}^{4}(x) = \begin{cases} \frac{1}{2}|x|^3 - |x|^2 + \frac{2}{3}, & |x| < 1 \\ -\frac{1}{6}|x|^3 + |x|^2 - 2|x| + \frac{4}{3}, & 1 \le |x| < 2 \\ 0 & \text{otherwise} \end{cases} \tag{2-13}$$

When $N \to \infty$, the limit of $h_{Bspline}^{N}(x)$ is the 0-mean Gaussian function

$$G(x,\sigma) = \frac{1}{\sqrt{2\pi\sigma}} e^{-\frac{x^2}{2\delta^2}}.$$

2.2.3 Gaussian Kernel

A family of interpolation kernels based on the Gaussian function is presented in [19]. These kernels are called Gaussian interpolation kernels.

Some of the one-dimensional Gaussian kernels are

$$h_N^2(x) = G^0(x,2\gamma_2) - \gamma_2 G^2(x,\gamma_2)$$

$$h_N^6(x) = G^0(x,2\gamma_6) - \gamma_6 G^2(x,\gamma_6) - \frac{\gamma_6^3}{24} G^6(x,\gamma_6) \qquad (2\text{-}14)$$

$$h_N^{10}(x) = G^0(x,2\gamma_{10}) - \gamma_{10} G^2(x,\gamma_{10}) - \frac{\gamma_{10}^3 G^6(x,\gamma_{10})}{24} - \frac{\gamma_{10}^5}{1920} G^{10}(x,\gamma_{10})$$

In (2-14), $G(x, \sigma)$ is a Gaussian function with zero mean and standard deviation σ, and $G^n(x, \sigma)$ is the nth derivative of $G(x, \sigma)$. The coefficients γ_2, γ_6 and γ_{10} in (2-14) are

$$\gamma_2 = \frac{1}{2\pi}(\frac{1}{\sqrt{2}}+1)^2, \quad \gamma_6 = \frac{1}{2\pi}(\frac{1}{\sqrt{2}}+\frac{39}{24})^2, \quad \gamma_{10} = \frac{1}{2\pi}(\frac{1}{\sqrt{2}}+\frac{4065}{1920})^2.$$

The region of support of the Gaussian kernel $h_N^M(x)$ is $(-N/2, N/2]$, which means the Gaussian interpolation uses the values of the nearest N grid pixels. M is the degree of the kernel. It is shown by [19] that only some of the kernels with even degrees exist. The first three existing Gaussian kernels have degree 2, 6 and 10, and are given in (2-14).

In the Fourier domain, the Gaussian kernel (see the analysis in [8]), especially those with high degrees, have an almost rectangular main lobe (note: the spectrum of the *sinc* function is ideally rectangular), and its side lobes attenuate very quickly. This property gives the Gaussian kernels the best concentration of energy, i.e., percentage of energy within the main lobe, among all kernels discussed in this chapter. The main lobe of the Gaussian kernel, however, is considerably wider than that of the *sinc* function, and therefore the Gaussian kernel is not a near perfect approximation of the *sinc* function.

2.2.4 Other Interpolation Kernels

Besides the methods reviewed in sections **2.2.1** through **2.2.3**, there are other members in the family of the traditional methods, such as the truncated *sinc*, the windowed *sinc*, and the Lagrange kernels ([8], [33]), etc. Due to their relatively rare usage and less practical importance, these methods are not reviewed here. For detailed discussion of these methods, refer to the corresponding references.

2.3 Transform-based Methods

Image interpolation can also be carried in the transform domain. Performing a transform on a digital image of H×V pixels yields a matrix of H×V of its transform coefficients. For the majority of real-life images, most energy is located in the low end of the coefficient matrix, and the high-end coefficients usually have very small values. By virtue of this property, the coefficient matrix of the interpolated image can be estimated from the coefficient matrix of the original digital image. The interpolated image is then obtained by performing the inverse transform.

Among the transform domain interpolation methods, the most frequently used is based on the Discrete Cosine Transform (DCT).

The procedure of the DCT-based interpolation is as follows.

(1) Perform DCT on the original digital image. This results in the image's transform matrix **Q**.

(2) Estimate $\mathbf{Q}_s$, the transform matrix of the interpolated image. For image reduction, $\mathbf{Q}_s$ is the top-left corner of matrix **Q** that has the same size as the reduced image.

For image expansion, $\mathbf{Q}_s$ is formed by copying $\mathbf{Q}$ to its top-left corner and assigning all remaining coefficients the value of zero.

The $\mathbf{Q}_s$ so obtained is then multiplied by the interpolation factor R to make the total energy constant.

(3) Perform the inverse transform on the coefficient matrix $\mathbf{Q}_s$. This yields the interpolated image.

Another transform domain interpolation method is based on the Wavelet Transform (WT). Wavelet transform splits the energy of the original image into the LL, LH, HL and HH bands. The coefficients in the LL band are used to estimate the coefficient matrix of the interpolated image. The WT coefficient matrix of the interpolated image is estimated through similar procedure as step (2) of the DCT-based method, and then the interpolated image is obtained by the inverse WT. The WT-based interpolation method has the restriction that the interpolation factor is 2^n, where n is an integer.

Following the similar idea described above, some researchers proposed interpolation methods based on transforms other than DCT and WT. Two examples are the DST (Discrete Sine Transform)–based method proposed in [69] and the FFT-based method proposed in [66].

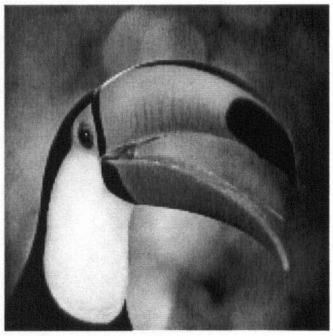
It is known that the transform-based interpolation methods suffer from certain degradations, e.g., the blockiness and ringing effects along image edges. Compared with the spatial domain interpolation methods, the transform-based methods have drawn less interest and practical usage. As a result, the transform-based methods are only briefly discussed, and will not be the focus of the remainder of this research.

2.4 Experimental Results of the Traditional Methods

In our experiments, we use some commonly used images as the original digital images, and perform interpolations on them using some of the traditional methods reviewed earlier. Here we show the results from two images: Camera Man and Tropic. The images are interpolated by replication, bi-linear, quadratic ($a=0.5$) and bi-cubic with $a=-0.5$ (Catmull-Rom). Camera Man (256×256) is expanded by a factor of 2×2. Tropic (320×320) is first reduced by the above methods by $\frac{1}{2}\times\frac{1}{2}$ and then expanded by 2×2, so that the interpolated image has the same size as the original.

Both images are gray-scale, with pixel value ranging from 0 to 255. The spatial resolution of display is chosen as 96 DPI. The original images are shown in Figure 2.8, and the interpolated images are shown in Figure 2.9 and 2.10.

The original image Camera Man



The reduced image Tropic

Figure 2.8 The test images

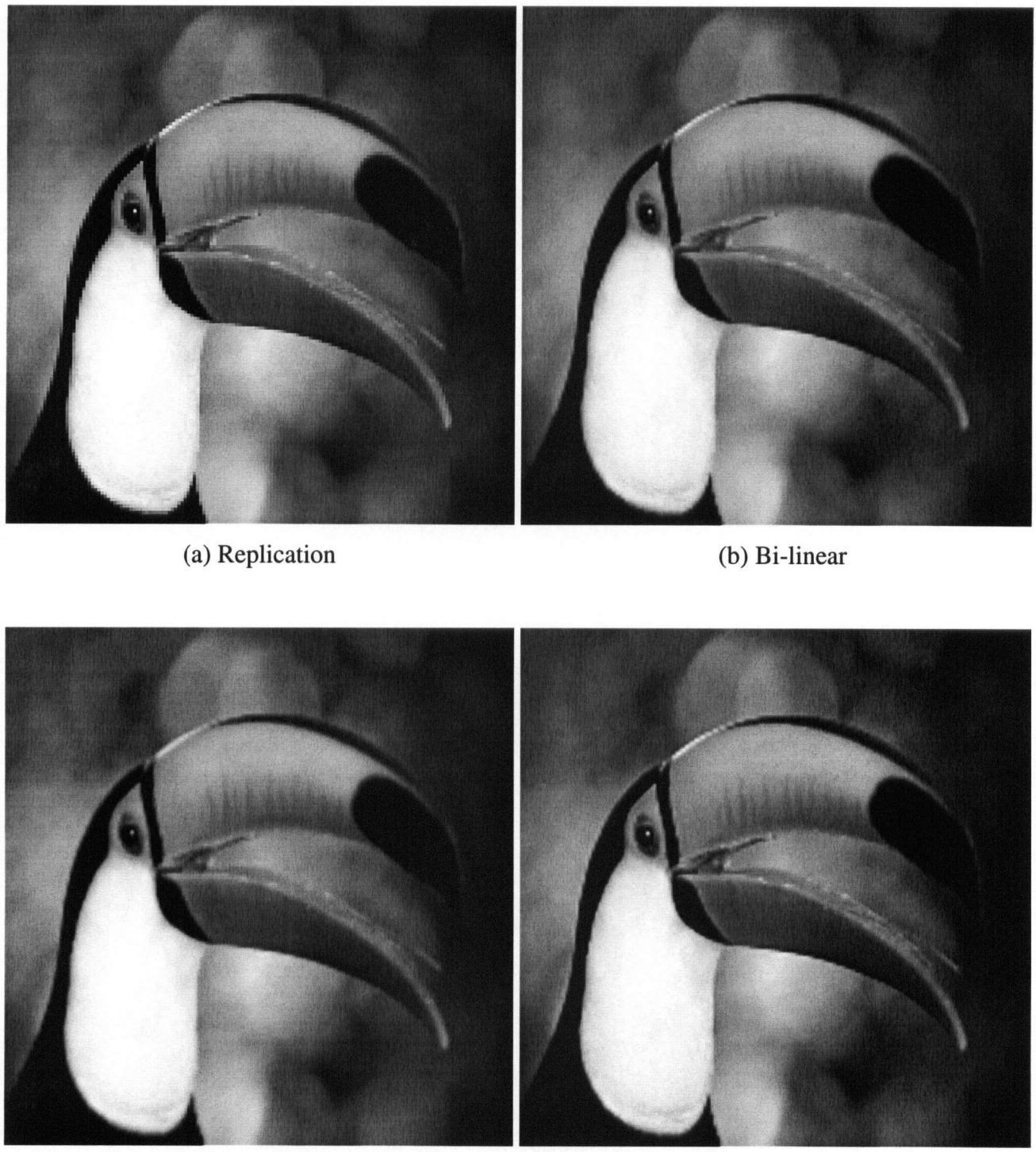
Image Tropic is reduced by $\frac{1}{2}\times\frac{1}{2}$.

Figure 2.9 (a) Expanded by pixel replication

Figure 2.9 (b) Expanded by bi-linear interpolation

Figure 2.9 (c) Expanded by quadratic interpolation with *a=0.5*

Figure 2.9 (d) Expanded by bi-cubic interpolation with *a= -0.5*



(a) Replication

(b) Bi-linear

(c) Quadratic with $a=0.5$

(d) Bi-cubic with $a= -0.5$

Figure 2.10 Tropic expanded by 2×2 after reduction

2.5 MSE-based Evaluation of Interpolation Methods

As discussed in section **2.1**, in a practical interpolation method, the original continuous image *f(x,y)* can only be approximately reconstructed. That is to say, in the general case an interpolation method always has errors. Meanwhile, the amount of error introduced by different methods varies. As a result, the qualities of the interpolated images are also different.

Several objective measures have been used by researchers for evaluating the performance of different interpolation methods. Among them, the Mean Squared Error (MSE) is one of the most commonly used.

The MSE is defined as

$$MSE = \sqrt{\frac{\sum_{m,n}(F(m,n) - F_d(m,n))^2}{N}} \tag{2-15}$$

in which $F_d(m,n)$ is the original digital image, $F(m,n)$ is the digital image resulting from interpolation, and N is the total number of pixels in the image.

The MSE measures are also often expressed in the form of Signal to Noise Ratio (*SNR*) or Peak Signal to Noise Ratio (*PSNR*). SNR and PSNR are respectively defined as

$$SNR\ (dB) = 10\log_{10}\left(\frac{\sum_{m,n} F_d(m,n)^2}{\sum_{m,n}\left(F(m,n) - F_d(m,n)\right)^2}\right) \tag{2-16}$$

and

$$PSNR\ (dB) = 10\log_{10}\left(\frac{D^2 \times N}{\sum_{m,n}\left(F(m,n) - F_d(m,n)\right)^2}\right) \qquad (2\text{-}17)$$

In (2-17) D is the maximum range of the pixel value. For the case of 8-bit gray-scale images, the value of D is 255.

In calculating *MSE*, *SNR* and *PSNR*, it is required that the interpolated image $F(m,n)$ has the same lattice as the original digital image $F_d(m,n)$. Since the sampling lattice and number of pixels are different before and after interpolation, (2-15), (2-16) and (2-17) cannot be directly calculated. Many researchers adopt an alternative approach to obtain *MSE*, *SNR* and *PSNR*, that is, to reduce the original digital image by a factor of $1/R$ (R>1) and then expand the reduced image by the factor of R, both through the same interpolation method. After this reduction-and-expansion process, the resulted image $F(m,n)$ will have the same lattice and number of pixels as the original image $F_d(m,n)$, and thus (2-15), (2-16) and (2-17) can be evaluated.

When evaluating the performance of a certain interpolation method using the above reduction-and-expansion approach, usually the reduction and expansion are carried using the same interpolation method. If different methods are used for reduction and expansion, the overall result is considered the combined performance of the two methods.

In Chapter 5, MSE of some traditional interpolation methods is evaluated and compared to the proposed interpolation method. For detailed results, see section **5.5.2**.

Chapter 3.
Recent Studies of Image Interpolation

In Section 1, we discuss the visual degradations commonly seen in most traditional methods. Section 2 describes the recent visually orientated interpolation methods, which focus on good visual quality of images. We survey the visually oriented interpolation methods by introducing their ideas and methodologies, and by examining some representative ones in details. Section 3 reviews the ***Edge-Directed*** and ***Orientation-Adaptive*** methods. Section 4 reviews the ***Level Set-based*** methods. In section 5, we summarize these newly introduced methods and their relationships, strengths and weaknesses are discussed

3.1 Artifacts in the Traditional Methods

At present, the majority of image interpolation methods are based on the traditional techniques, most of which are reviewed in Chapter 2. Among the traditional methods, the pixel replication, bi-linear and bi-cubic interpolation are the most widely used. These three methods are ubiquitous in the field of image interpolation, and have been used as the "standard" techniques by many researchers and applications developers.

Among the reasons why the traditional methods are so widely accepted, the most important one is their simplicity. Most traditional methods employ compactly supported kernels and simple algebraic operations that allow them to be implemented at low to moderate computational expense. The simplicity of the traditional methods makes them

applicable on almost all platforms with different computing capabilities, and especially suitable for applications that have real-time requirements.

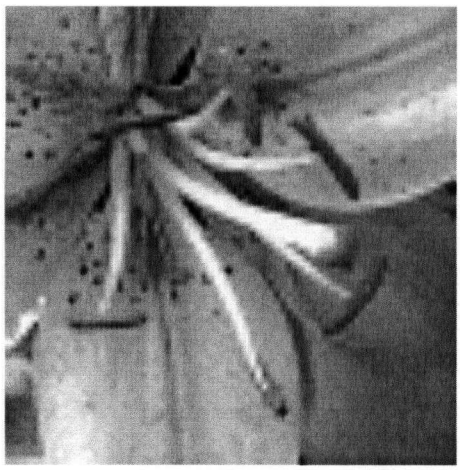
Despite the wide acceptance of the traditional methods, in most cases the qualities of the interpolated images they provide are less satisfying. The interpolated images usually suffer various types of degradations. Two of the worst degradations in the traditional methods are the ***zigzagging*** and the ***blurring*** artifacts, which are explained next.

3.1.1 Zigzagging Artifacts

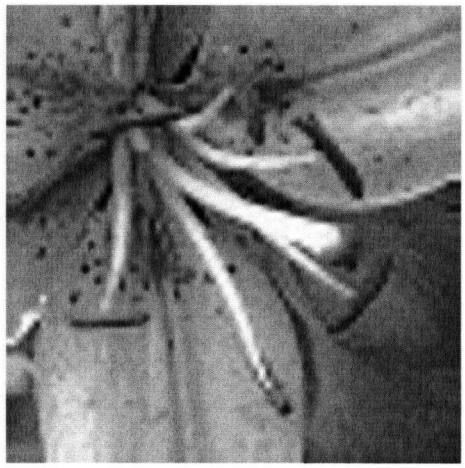
Zigzagging artifacts, sometimes referred to as aliasing errors or jaggies, are most commonly observed as staircasing of the image edges and thin lines, or moiré patterns in areas featuring fine texture. The zigzagging is a very annoying visual effect in that it makes the interpolated images look unnatural and poorly rendered. For a typical example of the zigzagging effects, see the interpolated images shown in Figure 2.8 and 2.9. In Figure 3.1 below, we show another example of zigzagging effects in image interpolation.

Almost all traditional interpolation methods suffer from the zigzagging artifacts, both for up-sampled and down-sampled images.

(a) The original image Flower (75×75)



(b) Through bi-linear interpolation



(c) Through bi-cubic interpolation

Figure 3.1 Flower expanded by 3×3

Notice the obvious zigzagging effects along the stamen and leaf edges.

3.1.2 Blurring Effects

Blurring effects only happen when an image is expanded (interpolation ratio $R > 1$). Blurring effects make the expanded images visually not as sharp as the original image, i.e., images suffer loss of contrast during the process of expansion. Usually the greater the expansion ratio, the more severe the blurring effects. As a result of blurring effects, especially when the expansion factor is large (e.g., 2.0 or greater), the expanded images are obviously fuzzy and visually not as pleasant. Blurring effects affect almost all traditional interpolation methods except the pixel replication. For an example of the blurring artifacts, see the expanded images in Figure 2.8, 2.9 and 3.1, especially the ones expanded by the quadratic method.

Note that blurring does not only happen in image interpolation. There are also other types of image blurring, such as motion blurring and the blurring resulting from out-of–

focus cameras. Within the scope of this research, only the blurring effect associated with image interpolation is of interest.

Apart from the examples we have shown above, visual artifacts are also commonly observed in printed media (e.g., newspapers), online images, digital videos, etc. In Figure 3.2 below, we show a real example of images visually degraded by the presence of zigzagging and blurring.

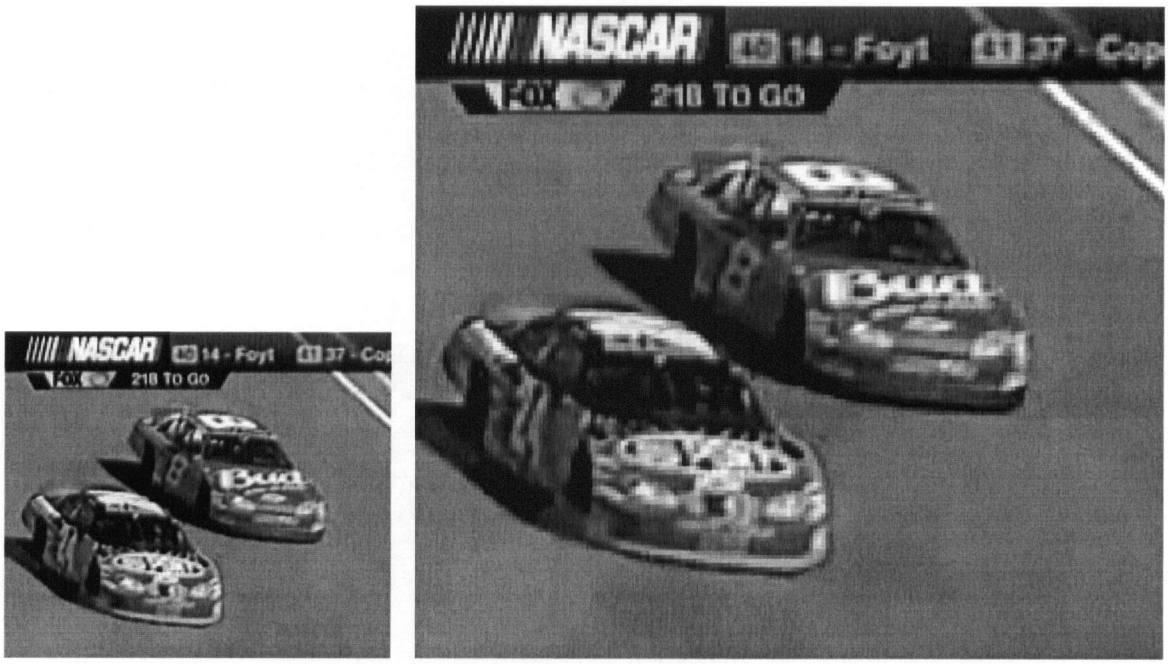


Figure 3.2 Visual artifacts in a real example.

Shown in the figure is a frame extracted from live Internet video stream played by RealPlayer®, a popular PC-based commercial video player. On the left is the frame at 100% scale, on the right is the same frame at 200% scale. The enlargement is provided by RealPlayer® itself. Notice the zigzagging effects along the edges of the cars, shadows and traffic lines.

Usually, blurring and zigzagging artifacts exist simultaneously in interpolated images. These artifacts, while affecting virtually every part of an interpolated image, are both

especially harmful to edges and image contours. Zigzagging artifacts are mostly harmful to expanded images, and blurring effects only happen in expanded images.

3.1.3 The MSE Evaluation and Visual Quality

In the traditional study of image interpolation, the performance of an interpolation method is usually evaluated by the MSE-based measurements (see section **2.5**). Accordingly, the emphasis of traditional interpolation is to achieve superior MSE measure in the interpolated images. Nevertheless, it is well realized that the MSE-based objective measures are often ineffective in evaluating the subjective quality of the interpolated images. These measures often fail to adequately represent the extent of an image being visually pleasing, i.e., the perceived quality.

As an example, the bi-cubic interpolation usually outperforms the bi-linear interpolation in MSE (for examples, see Table 7-1 and 7-2), but in many cases the subjective impression of the images it yields is not superior to that of the bi-linear interpolation (see the comparison in section **2.4** as an example). As another example, the bi-linear interpolation, which usually results in significantly better SNR than the pixel replication method, does not proportionally outperform the latter visually, largely due to its obvious blurring effect.

3.2 Visually Oriented Interpolation Methods

As a result of the disadvantages of the traditional methods, visually oriented interpolation has become the focus of recent studies on image interpolation. It is

commonly agreed by recent researchers that providing visually pleasant and artifact-free images should be the emphasis of the newly developed interpolation techniques. Accordingly, different new methods of image interpolation ([11]-[30]) have been proposed in recent years. These newly introduced methods include **edge-directed, direction-adaptive, data-dependant, contour preserving,** and **isophote-based** methods, and are distinguishingly different from the traditional methods. In our survey, these methods are categorized as "visually oriented" interpolation techniques.

Our survey shows that the methods of this new family generally have the following common features:

(1) They concentrate on the visual satisfaction of images. As discussed in the previous section, the presence of artifacts is an important reason why interpolated images suffer visual degradations. Expectedly, the elimination of artifacts is highly emphasized in almost all of the new methods.

(2) They emphasize visually pleasing reconstruction of important image features, the most important one being the edges. Because the subjective assessment of images is on its own a complicated topic, a comprehensive improvement in images' visual quality is difficult to achieve within a limited scope of study. As a result, most of the new methods currently focus mainly on improving image features that are already known to be crucial in image perception.

(3) Adopt content adaptive and non-linear techniques. In the traditional methods, only the *location* of the pixel to be interpolated and the *intensity value* of its nearest parent pixels are used in determining the value of an interpolated pixel. In the new methods, further content-related information, such as the location of edges, gradient

and curvature, is extracted from the original image, and used to obtain the interpolated pixel values.

Despite the similarities at the fundamental level as described above, the specific methodologies adopted by these new methods are highly diversified. Because of such diversity, it is not easy to establish a very clear classification of these methods. However, in our study, we approximately divide these methods into the following categories.

(1) **Edge-directed.**

(2) **Orientation-adaptive.**

(3) **Level Set-based.**

In the following sections, we discuss the above three categories of methods, by introducing their general approaches and examining their methodologies. It is not possible in this research to review with full detail all these methods. Instead, we only review a few representative ones from each category. Because of the relative closeness between the edge-directed and the orientation-adaptive methods, we review them in the same section.

3.3 Edge-Directed and Orientation-Adaptive Methods

Edges are the most prevalent features of images. Study of human visual system (HVS) shows that HVS is highly sensitive to image edges ([73], [74]), and this explains why edges are extremely important in the perception of images. As a result, almost all of the newer interpolation methods emphasize visually pleasing edges in the interpolated images.

It is accepted by many researchers that the quality of the interpolated images highly depends on two major factors: smoothness and sharpness of the edge ([11], [15]). These two factors correspond respectively to the elimination of the zigzagging and blurring artifacts on edges.

Some of the newly introduced methods ([11]-[14], [17]) employ explicit edge detection and manipulation in the interpolation process, and thus are referred to as the ***edge-directed*** methods; others ([15], [16], [18], [22]) use the edge information in alternative ways. These methods are referred to as ***orientation-adaptive***.

3.3.1 Edge-Directed Methods

The basic methodology in most of the edge-directed methods can be summarized as follows.

1. Detect (or estimate) the edges from the original digital image, and map the location of edges on the expanded lattice, or onto the continuous domain.
2. For children pixels near an edge, its value is obtained by a specially designed interpolation algorithm, which attempts to 1) preserve the contrast across the edge, and 2) prevent the interpolation from being carried across the edge.

   For pixels away from the edges, usually traditional interpolation is used to obtain their values.

Reference [11] proposes one of the earliest edge-directed interpolation methods. As an example, we briefly review this method as follows.

The basic idea in [11] is that when interpolating the value of a child pixel close to an edge, all parent pixels involved should be on the same side of the edge. An example is given in Figure 3.3.

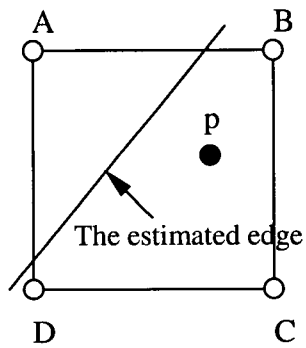


Figure 3.3 Edge-directed interpolation proposed by reference [11]

Assume A, B, C, and D are parent pixels, and *p* is a child pixel whose value is to be interpolated. If an edge line is detected going through block ABCD as shown in Figure 3.3, then the value of *p* is interpolated by parent pixels B, C, and D, which are on the same side of the edge. In [11], the interpolation of *p* using B, C and D is implemented as a bi-linear interpolation involving all parent pixels A, B, C and D, but with the value of pixel A (called the "corner pixel") replaced by an extrapolation through B, C and D.

To detect the edges, [11] employs an approximated Laplacian of Gaussian (LoG) filter. The edge is defined as the curve on which the output of the approximated LoG filter crosses 0. The interpolation algorithm (see Figure 3.3) requires location of edges with sub-pixel (fractional) precision, while the LoG filter cannot be directly applied on non-grid locations. Alternatively, the LoG filter is applied on the parent pixels, and the output for a non-grid point is approximately obtained by bi-linearly interpolating the output of the 4 nearest parent pixels.

Similar to [11], most of the edge-directed methods precisely locate edges with pixel or sub-pixel precision and very thin width through different techniques. The methods of edge estimation they employ and the corresponding precisions are listed in Table 3.1.

Among the edge-directed methods, besides the difference in edge detection, the approaches to interpolating the edge pixels are also largely different. Table 3.2 briefly summarizes their edge-interpolation methods. Note that some of the methods are restricted to expansion factors of integers, due to their specially designed mechanisms of interpolation.

Table 3-1. Edge estimation in different edge-directed methods

Method	**Edge Detection**	**Location of Edge**	**Width of Edge**
Ref. [11]	LoG, interpolated on non-grid points.	<1 pixel	<1 pixel
Ref. [12]	Model-based. Assumes that a continuous "step edge" exists, and solves its parameters from nearby original parent pixels.	<1 pixel	<1 pixel
Ref. [13]	Laplacian. Applied on grid pixels.	1 pixel	1 pixel
Ref. [17]	Canny. Applied on grid pixels.	1 pixel	1 pixel

Table 3-2. Edge interpolation in different edge-directed methods

Method	**Interpolation of Edge Pixels**	**Interpolation Ratio**	**Interpolation of Non-edge Pixels**
Ref. [11]	Modified bi-linear interpolation.	Arbitrary	Bi-linear
Ref. [12]	Estimate the continuous "step edge" in the continuous domain (1), and then re-sample it for near-edge children pixels.	2^n	N/A(2)
Ref. [13]	Model-based edge reconstruction. Edge line is chain-coded and mapped into expanded image.	Integers	Pixel replication
Ref. [17]	Gradient-based interpolation for pixels near edges.	Integers	Bi-linear

(1) See the corresponding entry of Table 3.1.

(2) Not specified by the reference.

3.3.2 Orientation-Adaptive Methods

The underlying assumption of the orientation-adaptive methods is that the interpolation should be performed along the orientation of the edges. This idea is similar to that of the edge-directed. On the other hand, the orientation-adaptive methods usually do not require very thin edges to be detected with high precision. Instead, the information of edges is used implicitly.

Among these methods, the methods by which interpolation is carried so as it adapts to the orientation, and their approaches largely differ. In the following we briefly review two of these methods.

The method proposed in [15] is based on the Wiener-filtering theory. The pixels of the interpolated image $F(m,n)$, according to the Wiener theory, can be predicted from the original digital image $F_d(m,n)$, if the covariance information of $F(m,n)$ is fully known. Because in general the covariance of $F(m,n)$ is unknown, just as $F(m,n)$ itself, [15] introduces an approach to estimate it from the covariance of the original image $F_d(m,n)$, which can be obtained from $F_d(m,n)$.

In [15] it is assumed that for areas near the edges, the image signal is stationary, so that $F(m, n)$ has approximately the same covariance characteristics as $F_d(m, n)$. In short, the covariance is "resolution-invariant" in the edge areas. Note that this assumption is NOT true for an image area with arbitrary nature, where the image signal cannot be unconditionally assumed stationary.

Based on the above assumption, the covariance matrix of $F(m,n)$ for the edge areas can be obtained by mapping the covariance matrix of $F_d(m,n)$ into the expanded lattice. Some approximations are made during the mapping. With the covariance of $F(m,n)$ (approximately) obtained, the unknown pixel values are calculated through the equations of Wiener filtering.

For non-edge areas, because the assumption that the covariance is resolution-invariant does not hold, the above approach is not applicable. Instead, [15] uses the traditional bi-linear method for the non-edge areas.

The algorithm takes the following 4 steps:

(1) Detect the edges in the original digital image, and map the edge areas into the expanded image.

(2) For edge pixels, calculate the covariance of the original image, and map it into the covariance matrix of the interpolated image after some approximations.

(3) For edge areas, children pixels are calculated using parent pixels and the covariance obtained from step (2), through Wiener filtering.

(4) For non-edge areas, bi-linear interpolation is performed.

The mapping mechanism of the covariance matrix imposes the restriction that this method is only applicable to interpolation ratio of 2^n (n is an integer). The authors suggest that arbitrary expansion ratio can be achieved by first expanding by the closest and greater 2^n and then down-sampling to the desired scale through bi-linear interpolation.

In the above method, the orientation information is embedded in the covariance matrix of the interpolated image and is used implicitly in the interpolation.

Reference [16] proposes another orientation-adaptive, PDE (partial differential equation)-based scheme. In this method the interpolation is treated as a variational problem — it seeks the reconstructed continuous image $\hat{f}(x, y)$ that satisfies certain conditions. In their method, the following two conditions are initially set:

(1) The magnitude of gradient is minimized over the whole image.

(2) The known (parent) pixel values remain unchanged after the interpolation (the Straightness property).

Reference [16] concludes that given the original digital image, finding the interpolated image $\hat{f}(x, y)$ by solving the partial differential equations yielded from the above two constraints is very difficult. Although an iterative numerical solution is possible, even disregarding its computational complexity, the result is often sensitive to the initial values. Note that this problem is encountered in the level set-based methods ([20] and [24]) reviewed in the next section.

In order to achieve a direct and one-pass solution of $\hat{f}(x, y)$, [16] suggests that another condition be added besides conditions (1) and (2) above. In their method, the third condition is chosen as:

(3) On the grid positions of $\hat{f}(x, y)$, the angle of the gradient has minimized error.

This condition can be interpreted as: $\hat{f}(x, y)$ has approximately the same direction of gradient on its grid_pixels as that of the original continuous image $f(x, y)$, i.e., $\hat{\theta}(m, n) \approx \theta(m, n)$.

Condition (3) is another variational problem by itself, which is solved in [16] by making some approximations. It is then shown in [16] that under conditions (1), (2) and (3), the interpolated image $F(m,n)$ can be directly calculated from the original digital image.

By adding condition (3), the orientation information is introduced into the interpolation process through a series of derivations. In this method, the edges are not directly detected, but the orientation of gradient is estimated and used in the interpolation process. Therefore we categorize [16] as an orientation-adaptive method.

3.4 Level Set-Based Methods

The ***Level Set Methods*** are a set of methodologies originally introduced by Sethian, Osher et al ([40], [42], [43]), studying the problem of dynamic contours (also known as the propagating fronts, or Snakes).

In the Level Set Methods, the curve (contours) to be studied is considered as the level set contour (i.e., the equi-intensity contour) of a certain surface, which is called the ***level set function***. By doing so, the analysis of the curves can be equivalently replaced by that of their level set function. With the help of the methodologies of differential geometry, the Level Set Methods discusses how the dynamic contours behave under given circumstances, and more importantly, how to control their behaviors (such as change of shape, speed of motion, etc) in order to make them satisfy certain requirements. The Level Set Methods have found applications in different areas of image processing, including shape evolution, de-noising, object extraction, etc.

It is not until recently that interpolation methods based on the level set theories were proposed in [20], [24] and [29]. These methods are reviewed below.

In the methods proposed in [20] and [24], the zigzagging artifacts are reduced by smoothing the isophotes of the interpolated image. Their interpolation method adopts a variational approach, in which the interpolated image $\hat{f}(x, y)$ is required to meet the following constraints.

(1) In the reconstructed continuous image $\hat{f}(x, y)$, the curvature of the isophotes is minimized.

(2) The values of the parent pixels are unchanged after the interpolation (Straightness).

Starting from the above two restrictions, derivations from the level set theory show that such $\hat{f}(x, y)$ must satisfy the condition

$$\Delta\hat{f}(x, y) = -c(x, y) \cdot \left\| \nabla\hat{f}(x, y) \right\| \qquad (3\text{-}1)$$

where $\Delta\hat{f}(x, y)$ is the total differential of $\hat{f}(x, y)$, *c(x, y)* is the curvature of the isophote at point *(x, y)*, and ∇ is the gradient operator. Condition (3-1) is interpreted as: the (differential) change of $\hat{f}(x, y)$ at point *(x, y)* should equal the product of the local curvature and the magnitude of the gradient; or, each point of the isophote should move in the normal direction with the speed proportional to the curvature of that point.

Unfortunately, the analytical solution of equation (3-1), in a general case, is very complicated. Alternatively, Morse et al adopted an iterative numerical approach that yields an approximate solution for equation (3-1). The procedure of their method is explained as follows.

Step 1. Interpolate the original digital image $F_d(m, n)$ by a known method, say bi-cubic, resulting in an initially interpolated image $F^{(1)}(i, j)$.

Step 2. Calculate curvature $c(i, j)$ for all pixels of $F^{(1)}(i, j)$. To do this, the derivatives of the original image $f(x, y)$ are needed, and they are estimated from $F^{(1)}(i, j)$.

Step 3. Modify $F^{(1)}(i, j)$ by adding an increment $\Delta^{(2)}(i, j)$ to it, resulting in an updated version of the interpolated image, $F^{(2)}(i, j)$, that is,

$$F^{(2)}(i, j) = F^{(1)}(i, j) + \Delta^{(2)}(i, j)$$

in which

$$\Delta^{(2)}(i, j) = \begin{cases} -c(i, j) \cdot \left|\vec{G}(i, j)\right|, & \text{if (i, j) is a child pixel} \\ 0 & \text{if (i, j) is a parent pixel.} \end{cases}$$

Curvature $c(i, j)$ and $\vec{G}(i, j)$, the gradient of pixel (i, j), are estimated in Step 2.

Step 4. Repeat Step 2 and 3.

During the kth iteration, the $F^{(k-1)}(i, j)$ is used to estimate the curvature and gradient, which in turn decide the next increment $\Delta^{(k)}(i, j)$.

Step 5. If, after the nth iteration, $\Delta^{(n+1)}(i, j)$ is found to have very small values, the iteration is considered to have converged and thus stopped. The image resulting from the last iteration, $F^{(n)}(i,j)$, is the final interpolated image, that is

$$F(i, j) = F^{(n)}(i, j)$$

See Figure 3.4 for the flow diagram of the level set-based method. Experiments in [20] and [24] show that their level set-based interpolation method reduces the zigzagging artifacts in the interpolated images.

Another level set-based method is proposed by Aly and Dubois in [29]. In this method, the reconstructed continuous image $\hat{f}(x, y)$ is required to satisfy a data fidelity criterion, which is derived from the observation model proposed in [29]. It is pointed out in [29] that obtaining $\hat{f}(x, y)$ by solving the yielded partial differential equation is an ill-posed problem, which often results in undesired noise and artifacts in the interpolated images. To solve this problem, an extra constraint that the gradient of $\hat{f}(x, y)$ is minimized is added in [29].

Combining the data fidelity condition and the minimum-gradient condition, and through derivations of the Level Set Method, an iterative solution of the interpolated image *F(m, n)* is obtained in [29].

Experimental results in [20], [24] and [29] show that the level set-based methods yield less artifacts and improved visual quality in the interpolated images, compared with the traditional methods.

3.5 Discussions and Summary

In Section **3.3** and **3.4**, we reviewed the recently introduced interpolation methods, which are all visually oriented and content-adaptive. Overall, these methods have shown that they are effective in achieving improved visual qualities. The new methods outperform the traditional methods by providing interpolated images that are more visually pleasing and subject to less degradation. As a result, the newly emerging visually oriented techniques have been attracting much research interest.

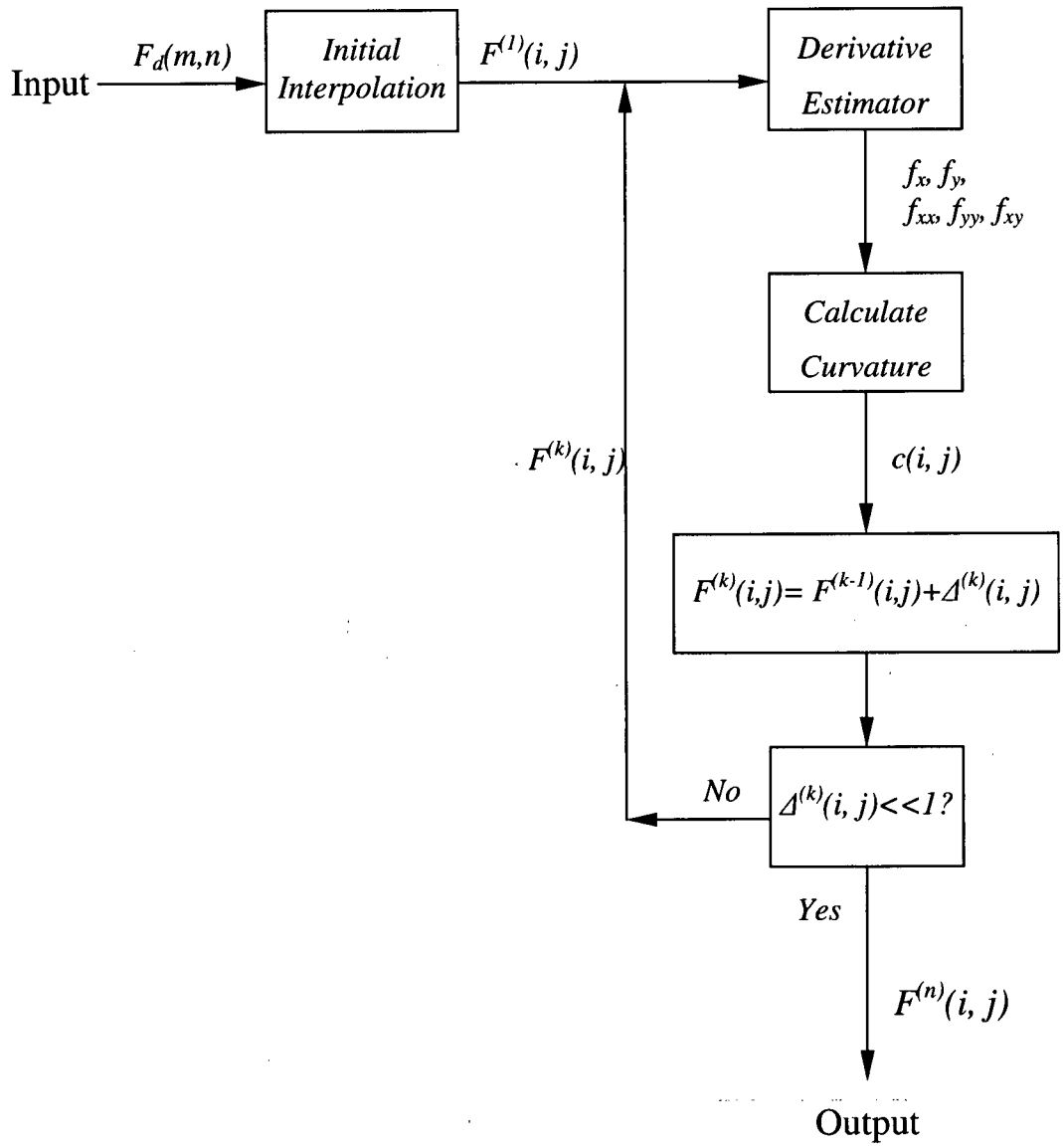


Figure 3.4 Flow diagram of the interpolation method proposed in [20] and [24]

Based on our survey, we summarize the logical relationship between the visually oriented methods, and then discuss the advantages and disadvantages of these methods.

- Among members of the visually oriented family, the edge-directed methods are the first to be developed. Their emphasis on visually pleasing edges is acknowledged by the later studies;

- The orientation-adaptive methods broaden the idea of "edge-directed" to "orientation-directed", and overcome some disadvantages of the edge-directed methods;
- The level set-based methods introduce the isophote-orientated approaches, in which the concept "orientation" can be strictly defined, and the cause of the zigzagging artifact can be more clearly explained.

The edge-directed methods emphasize the importance of image edges and employ specially designed, non-linear techniques to reconstruct more visually pleasing edges. Despite their obvious contributions in developing the family of new interpolation methods, some edge-directed methods still have the following disadvantages:

(1) These methods require edges to be "pinpointed", and in turn their performance is often dependent on obtaining edge estimators with high precision. Such edge detectors are usually computational costly.

(2) In these methods the edges are often defined as very thin lines segmenting different parts of the image. Pixels on and off the edge lines are treated differently. This rigid definition of edges in turn introduces strong non-linearity into the interpolation process, which results in some undesired artifacts. One of the so resulting artifact is referred to as the "cartooning effect". This disadvantage is also mentioned in [20] in a slightly different manner.

Comparing the edge-directed methods with the orientation-adaptive methods, the later are more flexible and linear, in the sense that they attempt to employ orientation-adaptive linear interpolators. Also, the orientation-adaptive methods do not emphasize the location of the edges with high precision, but rather their orientations. The orientation-adaptive

methods can be considered as a generalization of the edge-directed methods, with its same basic idea inherited.

The concept "orientation-adaptive" is further clarified by the introduction of isophote-oriented methodologies ([20], [24], [29], [39]). The isophotes, as will be discussed in full details in the next chapter, are image features on which orientation can be clearly defined and the zigzagging artifacts can be properly explained.

The level set-based methods are among the first few methods to suggest that the zigzagging artifacts should be studied from the isophote point of view. Introduction of isophote-orientated approach is the primary advantage of the level set-based methods. As we will show in the next two chapters, zigzagging artifacts can be effectively analyzed in the isophote field, and through this analysis isophote-oriented interpolation methods that are superior in providing zigzagging-free images can be developed,

However, the level set-based methods still have some disadvantages. Some of these disadvantages are listed as follows.

(1) High computational expense caused by the iterative process. The computational cost, especially when the interpolation factor is large, is a serious concern.

(2) The results sometimes "depend heavily on the initial approximation" ([24]), that is, with different methods chosen for the initial interpolation method, the eventual interpolated image could be considerably different.

(3) In these methods, the isophotes are manipulated implicitly and numerically. Although this does not affect their underlying smoothing of the isophotes, the isophotes are not shown in an explicit and visual way.

Our study also shows that many of the new interpolation methods have one common drawback, that is, the blurring artifact is not sufficiently addressed except in [12], [25] and [29]. In most of the methods reviewed in this chapter, the blurring and zigzagging artifacts are not clearly differentiated and dealt with accordingly. Instead, they are often inadequately mentioned in general terms, e.g., the artifacts along the edges, or visual degradation on edges. All the methods except [12], [25] and [29] either do not propose a solution for the blurring effects, or do not focus on this problem within their scopes.

Also, many of the newly proposed methods suffer from the following two limitations:

(1) The interpolation ratio is often limited to integers or 2^n. In many cases, such limitation is a result of a specially designed mechanism in the interpolation method, for example, the mapping between the original and the expanded grids.

(2) The interpolation process is often multi-pass or iterative. This disadvantage is associated with most of the PDE-based methods (except [16]) and some non-PDE-based methods.

Based on an examination and analysis of the different types of recent methods, we reached the conclusion that the strengths of the visually oriented methods could be further explored and improved by studying the content-adaptive interpolation from the isophote point of view. And to do this, we find it necessary to first study the behavior of isophotes in image interpolation and the performance of different methods in the interpolated isophotes. As a result, in the next chapter, we continue our research of visually oriented interpolation by studying the isophote fields of interpolated images.

Chapter 4.
Isophote Analysis in Image Interpolation

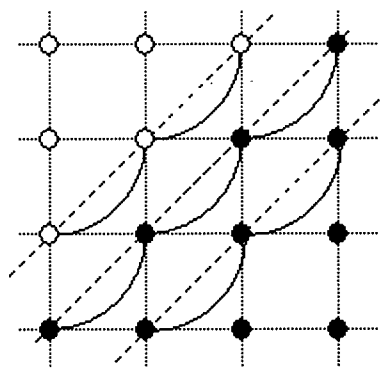
In this chapter, we study the isophotes of interpolated images, in order to understand the nature of zigzagging artifacts and why they arise. To do this, we first discuss the relationship between image interpolation and isophote reconstruction. Then we derive the form of isophotes of images interpolated by the bi-linear and bi-cubic methods. The curvature of isophotes in bi-linear interpolation is then analyzed.

4.1 Zigzagging Artifact and Image Isophote

As discussed in Chapter 3, the visually oriented interpolation methods have shown encouraging effectiveness in removing zigzagging artifacts associated with traditional interpolation methods. It is suggested by recent researchers ([20], [38]) that zigzagging effects should be studied and solved by studying isophotes of the interpolated images. Correspondingly, some isophote-oriented interpolation methods have been introduced, among which are the level set-based methods reviewed in Chapter 3. In these methods zigzagging artifacts are explained as a result of excessive curvature in isophotes of the interpolated images.

The formal definition of an isophote and further discussions on isophotes are given in the following section. We show later in this chapter, that after interpolation smooth

isophotes of the original image become piecewise curves, regardless of their real shape in the original image. These piecewise curves have similar patterns in their curvature changes, which are repeated along the image contours. In the intensity domain, these psuedo-periodic patterns of curvature are visually perceived as zigzagged contours. Figure 4.1 illustrates an example of what may happen to isophotes in an area containing a straight edge after traditional bi-linear interpolation.



Small circles: Original (known) pixels
– – – – – Original isophotes
——— Isophotes after bi-interpolation.

Figure 4.1 Illustration of zigzagged isophotes after bi-linear interpolation

Although the importance of isophotes in image interpolation has been noticed by some researchers, and some implicit isophote-oriented interpolation methods have been proposed ([20], [24], [29]), not much effort has been made to study the performance of different interpolation methods by analyzing the resulting isophotes. The linkage between zigzagging effects and distorted isophotes (illustrated in Figure 4.1) has not been shown analytically. Also, not many isophote-based measurements of the image quality have been introduced to measure the artifacts in interpolated images. As a result, a direct and

explicit study of the interpolation methods so as to explain what happens to isophotes is still to be sought, and the strength of isophote-oriented interpolation methods remains to be fully explored. It is noticed that some academic software packages do provide functions for approximately displaying isophotes of digital images, such as the CONTOUR function of MATLAB®. However, these functions do not have sub-pixel spatial precision, i.e., they do not show the true curve of isophotes between sampled pixels. Therefore these functions cannot be used as exact tools when studying the jaggies in image interpolation.

We first study the isophote field of interpolated images. In the following sections of this chapter, we first discuss the relationship between image interpolation and isophote reconstruction, and then show how to find the isophotes in bi-linear and bi-cubic interpolated images.

4.2 Interpolation and Isophote Reconstruction

Isophotes, also known as equi-intensity contours or level set contours, are significant features in the perception of images. One of the most important visual features, the image edge, is where isophotes are (near) parallel and densely located. Isophotes have already been extensively used in many other research areas such as astronomy.

An isophote is defined as the curve in the continuous x-y plane on which all points have the same intensity value. For simplicity, the isophote of a particular intensity value I_0 is referred to as isophote I_0. By virtue of its definition, the curve of isophote I_0 of a continuous function $f(x,y)$ can be found by solving the equation $f(x,y)=I_0$. The set of all

isophotes of $f(x,y)$ is called the isophote field or isophote domain of $f(x,y)$. The isophote field forms another representation of a continuous image $f(x,y)$.

Despite the virtual equivalence between a continuous function $f(x,y)$ and its isophote representation, given an arbitrary $f(x, y)$, its isophote field is not always known. This is because the explicit relationship between x and y is not always obtainable by analytically solving equation $f(x,y)=I_0$.

After sampling, the intensity information of the continuous image $f(x,y)$ is partially lost, with the only known intensity values being on the sampled pixels. Meanwhile, much information in the original isophotes field is also lost — only isophote points on the grid pixels are known, and the curve shapes connecting the grid points are unknown.

As discussed in section **2.1**, in image interpolation, the estimate of the original continuous image $\hat{f}(x, y) \approx f(x,y)$ is obtained by

$$\hat{f}(x, y) = \sum_m \sum_n F_d(m,n) \cdot h(x, y)$$

We re-write $\hat{f}(x, y)$ as a function of the sampled values and spatial variables x and y, i.e.,

$$\hat{f}(x, y) = \Phi\left[x, y, F_d(m,n)\right], \quad \text{for all } (m,n) \in F_d(m,n) \qquad (4\text{-}1)$$

In (4-1) $\Phi(.)$ is the interpolation operator.

Given an intensity value I_0 and letting $\hat{f}(x, y) = I_0$, (4-1) yields

$$I_0 = \Phi\left[x, y, F_d(m,n)\right] \qquad (4\text{-}2)$$

By definition, the curve described by (4-2) is the implicit form of isophote I_0 of the

reconstructed image $\hat{f}(x, y)$. If for every I_0, the solution of

$$I_0 - \Phi[x, y, F_d(m,n)] = 0 \qquad (4\text{-}3)$$

yields an explicit relation between x and y for any possible value of I_0, then the explicit form of the isophote field of the reconstructed image $\hat{f}(x, y)$ can be obtained.

Equation (4-3) shows the fact that, given $F_d(m,n)$, the isophote of the reconstructed image $\hat{f}(x, y)$ is determined by the interpolation operator $\Phi(.)$. That is to say, *an interpolation method reconstructs the original image in the intensity domain; it also reconstructs the isophotes of f(x,y).*

Each interpolation method has its unique operator $\Phi(.)$. Given $F_d(m,n)$, each method yields an $\hat{f}(x, y)$ that is different from other methods, and thus the interpolated isophotes are different, i.e., *each interpolation method has its unique way of reconstructing the isophotes of the original image.*

Because of its mathematical complexity, for an arbitrary $\Phi(.)$, (4-3) usually is not analytically solvable. Yet we show in the following sections that for the bi-linear and bi-cubic interpolations, (4-3) can be solved as polynomial equations, and thus the explicit form of the reconstructed isophote associated with these interpolation methods can be obtained.

4.3 Isophote Reconstruction by Different Interpolation Methods

4.3.1 Bi-linear Interpolation

Consider a pixel block formed of four adjacent parent pixels (0,0), (1,0), (1,1) and (0,1) (see Figure 4.2). The intensity values are only known at these pixels and we denote them as A, B, C and D. Assume the intensity value $\hat{f}(x, y)$ of a pixel *P(x,y)* lying inside this block is to be interpolated.

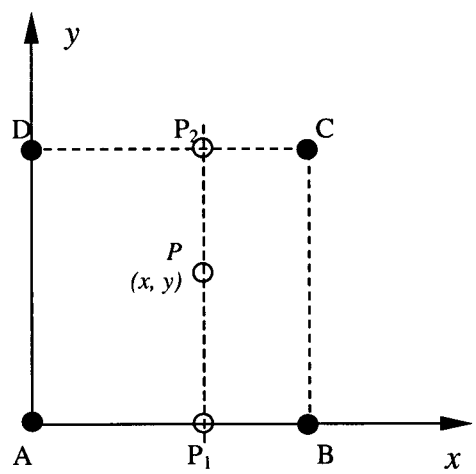


Figure 4.2 Bi-linear interpolation

In bi-linear interpolation, the value of the interpolated pixel P is obtained by

$$\hat{f}(x, y) = P_1 \cdot (1 - y) + P_2 \cdot y \qquad (4\text{-}4)$$

where pixels P1 and P2 have the same horizontal coordinates as point *(x,y)*, and their values can be preliminarily interpolated as

$$P_1 = A \cdot (1 - x) + B \cdot x, \quad P_2 = D \cdot (1 - x) + C \cdot x \qquad (4\text{-}5)$$

Substituting (4-5) into (4-4) and letting

$$\alpha = B - A,\ \beta = D - A,\ \gamma = A + C - B - D$$

(4-4) becomes

$$\hat{f}(x, y) = A + \alpha\, x + \beta\, y + \gamma\, xy \qquad (4\text{-}6)$$

To obtain the isophote of intensity value I_0 inside the ABCD block, we let $\hat{f}(x, y) = I_0$ in (4-6) and solve the resulting equation

$$A + \alpha\, x + \beta\, y + \gamma\, xy - I_0 = 0 \qquad (4\text{-}7)$$

with y as the unknown. The solution, which is given in (4-8), is the isophote I_0, i.e. the curve of all points (x, y) having the constant intensity value I_0, within block ABCD.

$$y = \frac{\gamma\,(I_0 - A) + \alpha\,\beta}{\gamma^2\,(x + \frac{\beta}{\gamma}\,)} - \frac{\alpha}{\gamma}, \qquad \gamma \neq 0 \qquad (4\text{-}8)$$

The x and y in (4-8) must satisfy the restriction that point (x, y) lies inside the block ABCD, i.e.

$$0 \leq x,\ y \leq 1 \qquad (4\text{-}9)$$

From (4-8) it is seen that in each block ABCD, any isophote I_0 takes the form of a hyperbola, determined by parameters α, β and γ. Thus within a grid block, the traditional bi-linear interpolation always implicitly uses a hyperbola to approximate an original isophote irrespective of the real shape of the original isophote (see Figure 4.1). For different blocks, α, β and γ assume different values that yield different hyperbolae. Therefore, the entire isophote I_0 is a curve of piece-wise hyperbolae.

4.3.2 Bi-cubic Interpolation

In the 4-point bi-cubic interpolation (see Figure 4.3), the interpolated pixel value is obtained by

$$\hat{f}(x,y)=\sum_{m=-1}^{2}\sum_{n=-1}^{2}P_{n,m}\,r(x-m)\,r(y-n),\ 0\le x,y<1 \qquad (4\text{-}10)$$

in which $P_{n,m}$ is the pixel value at grid point *(n, m)* and *r(x)* is the cubic interpolation functions discussed in Chapter 2 (refer to **2.2.1**).

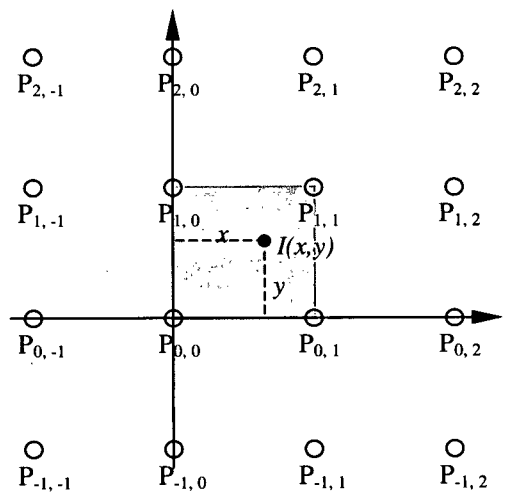


Figure 4.3 Bi-cubic interpolation

In our analysis, we use the Keys' cubic function as the interpolation function in formula (4-10). Its one-dimensional general form is

$$r(x)=\begin{cases}(a+2)|x|^3-(a+3)|x|^2+1 & 0\le|x|\le1\\ a|x|^3-5a|x|^2+8a|x|-4a & 1\le|x|\le2\\ 0 & |x|>2\end{cases} \qquad (4\text{-}11)$$

In order to solve equation $\hat{f}(x,y)=I_0$, (4-10) is converted into a cubic equation with variable y as the unknown. In doing so we substitute (4-11) into (4-10), and then

expand (4-10) into the sum of 256 terms of power products of x and y, we then merge the terms with the same order, and (4-10) becomes

$$\hat{f}(x, y) = \sum_{n=0}^{3}\sum_{m=0}^{3}\beta_{n,m}x^n y^m \quad 0 \le x, y < 1 \tag{4-12}$$

in which $\beta_{n,m}$ is the coefficient of the corresponding $x^n y^m$ term. $\beta_{n,m}$ is given by

$$\beta_{n,m} = \mathbf{P} \bullet \mathbf{E}_{4\times(3-n)+(3-m)+1} \tag{4-13}$$

In (4-13), $\boldsymbol{P}$ is the row vector formed by the parent pixel intensity values, i.e.,

$$\mathbf{P} = [P_{2,-1}\ P_{2,0}\ P_{2,1}\ P_{2,2}\ P_{1,-1}\ P_{1,0}\ \ P_{1,1}\ P_{1,2}\ P_{0,-1}\ P_{0,0}\ P_{0,1}\ P_{0,2}\ P_{-1,-1}\ P_{-1,0}\ P_{-1,1}\ P_{-1,2}] \tag{4-13b}$$

$\boldsymbol{E}$ is a 16×16 coefficient matrix resulting from the expansion of (4-10). In (4-13) we use $\boldsymbol{E}_k$ to represent the kth column of matrix $\boldsymbol{E}$. See appendix **A-1** for matrix $\boldsymbol{E}$.

We consider (4-12) a cubic equation with y the unknown, that is,

$$\hat{f}(x, y) = c_3 y^3 + c_2 y^2 + c_1 y + c_0 \tag{4-14}$$

in which

$$c_m = \sum_{n=0}^{3}\beta_{n,m}\cdot x^n \tag{4-15}$$

For a given x, the coefficients $c_3 \sim c_0$ in equation (4-14) are fixed, and given a pixel value $I(x,y)= I_0$, (4-14) becomes

$$c_3\ y^3 + c_2\ y^2 + c_1\ y + c_0 - I_0 = 0 \tag{4-16}$$

By solving cubic equation (4-16) and pairing the corresponding root y with the free variable x, a series of points (x, y) can be found, which form the isophote I_0 within the local interpolation block (the shaded area in Figure 4.3).

We rewrite (4-16) it as

$$y^3 + b_2 y^2 + b_1 y + b_0 = 0 \tag{4-17}$$

and use the method given by [60] to solve (4-17) as follows.

Let

$$p = b_1 - \frac{b_2^2}{3} \text{ and } q = \frac{-2b_2^3}{27} + \frac{b_1 b_2}{3} - b_0$$

The root(s) of (4-17) are then obtained by discussing the following cases.

Case I $p = 0$. There is one real root

$$y_i = \sqrt[3]{q} - \frac{b_2}{3}$$

Case II $p \neq 0$. There are four further sub-cases II(a), II(b), II(c) and II(d). In the first three sub-cases, there is one real root, and in sub-case II(d) there are three real roots. To discuss these sub-cases, define

$$h = \sqrt{\frac{4|p|}{3}}, k = \frac{3}{h|p|}, \text{ and } C = q * k$$

In all four sub-cases, the real root(s) have the form

$$y_i = h * z_0 - \frac{b_2}{3} \qquad (4\text{-}18)$$

In (4-18) z_0 is an interim unknown (in the three-real-root case, y_i and z_0 both represent three values). z_0 is determined by the value of p and C.

Sub-case II (a) $p > 0$, there is one real root y_i whose value is determined by (4-18) and

$$z_0 = \sinh[\frac{1}{3}\sinh^{-1}(C)] \qquad (4\text{-}19a)$$

Sub-case II (b) $p < 0$ and $C \geq 1$, there is one real root, and

$$z_0 = \cosh[\frac{1}{3}\cosh^{-1}(C)] \qquad (4\text{-}19b)$$

Sub-case II (c) $p < 0$ and $C \leq -1$, there is one real root, and

$$z_0 = -\cosh[\frac{1}{3}\cosh^{-1}(-C)] \qquad (4\text{-}19c)$$

Sub-case II (d) p < 0 and |C| ≤ 1, there are three real roots $y_i^{(1)}$, $y_i^{(2)}$ and $y_i^{(3)}$ related to $z_0^{(1)}$, $z_0^{(2)}$ and $z_0^{(3)}$ correspondingly, and

$$z_0^{(1)} = \cos(T), \quad z_0^{(2)} = \cos(T - \frac{2\pi}{3}), \quad z_0^{(3)} = \cos(T + \frac{2\pi}{3}) \qquad (4\text{-}19d)$$

in which

$$T = \frac{1}{3}\cos^{-1}(C) \qquad (4\text{-}19e)$$

The *sinh(x)* and *cosh(x)* in (4-19) are the hyperbolic sine and hyperbolic cosine functions, respectively.

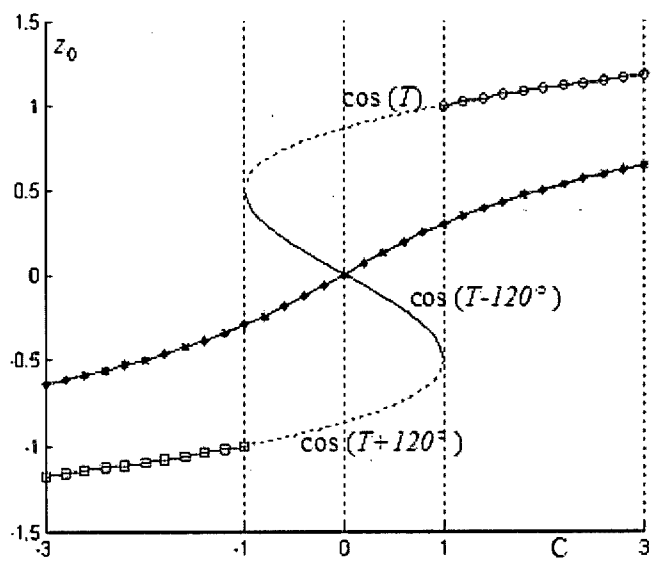


Figure 4.4 The z_0 — C curves, all four sub-cases.

*-marked:	sub-case II (a).
circle-marked:	sub-case II (b).
square-marked:	sub-case II (c).
upper dashed line:	sub-case II (d), $z_0^{(1)}$.
solid line:	sub-case II (d), $z_0^{(2)}$.
lower dashed line:	sub-case II (d), $z_0^{(3)}$.

For equation (4-16), explicit solutions are obtained by (4-13) through (4-19). For each given value of the free variable x_i, one or three y_i can be found and the corresponding

isophote point(s) are located. However, because of the complication of solving the cubic equations, in the general case there is no concise and straightforward relationship between x and y. In sub-case II (see equations (4-19)), the interim unknown z_0 is a function of the variable C, which is furthermore and eventually a function of x. As a part of the complicated x-y relationship, the $z_0 - C$ curve is drawn in Figure 4.4, illustrating how z_0 varies with C in sub-case II (a) through II (d).

As seen from Figure 4.4, for $-1<C<1$ (sub-case II (d)), the three z_0's are located on three types of curves, $cos(T)$, $cos(T\text{-}120°)$ and $cos(T+120°)$, respectively. For each of the three real roots $y_0^{(1)}$, $y_0^{(2)}$, $y_0^{(3)}$, we assign a type to it, according to the type of its corresponding z_0. For convenience, we write the three types as T, T-120 and T+120, respectively.

4.3.3 Tracking and Displaying the Isophotes

By solving equation (4-8) or (4-16), we are able to find point (x_i, y_i) on isophote I_0 in each interpolation block for each given horizontal coordinate x_i. When visualizing the isophotes, as x is a continuous variable, we cannot calculate all points (x, y) of an isophote. In order to display the curve of isophote I_0, we find some of the (x_i, y_i) points of isophote I_0 and then join these points by straight-line segments, i.e., we link the obtained discrete isophote points to form the reconstructed continuous isophote.

In our experiments, we calculate points of isophote I_0 with equally spaced x_i. In order to do so, we set a series of grid points on the x (horizontal) axis with spacing Δ_t. Δ_t is defined as the spatial resolution of the reconstructed isophote.

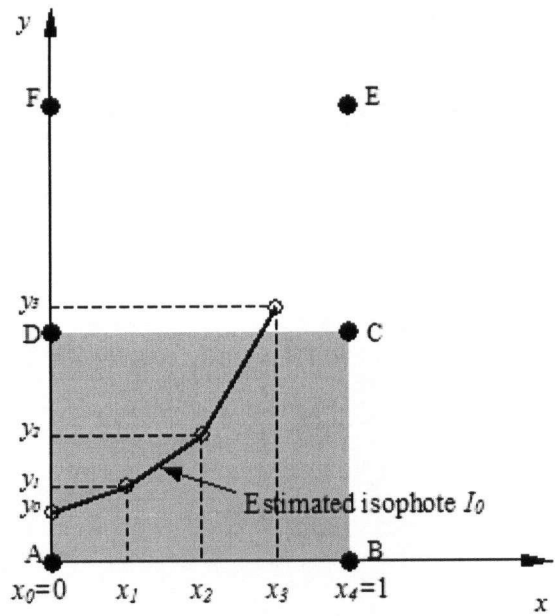


Figure 4.5 Tracking the isophote

In Figure 4.5, x_3 yields a $y_3 > 1$, indicating that isophote I_0 goes out of the upper bound of the current interpolation block enclosed by ABCD, and such y_3 is invalid. As a result, y_3 is re-calculated using parent pixels DCEF, which yield a valid new value of y_3.

The whole image is formed of a number of interpolation blocks with each block formed by four (bi-linear) or sixteen (bi-cubic) parent pixels surrounding it. We solve equation (4-8) or (4-16) for every one of these blocks in the test image and this yields all points of isophote I_0 horizontally positioned on the grid with spacing Δ_t.

On each grid position $x_i = i*\Delta_t$ (i is integer), the corresponding y_i is found as described in **4.3.1** and **4.3.2**. Each yielded solution point (x_i, y_i) must satisfy (4-9) to be a valid point on isophote I_0 in the current interpolation block, otherwise this point is discarded.

To display the curve of isophote, the discrete points found in different blocks have to be joined. We track the isophote by monitoring the yielded solution y_i's in each block. $y_i > 1$ indicates that the isophote goes to the upper block so the current segment of the

isophote should be linked to that in the upper block; similarly y_i <0 indicates linkage with the lower block. If all y_i's in a block are between 0 and 1, then the isophote goes to either the left or the right block. Refer to Figure 4.5 for an example of tracking an isophote in bi-linear interpolation.

In bi-cubic interpolation, solving (4-16) with a given x_i sometimes results in more than one (up to three) y_i's with values between 0 and 1, which means more than one isophotes of I_0 exist. In this case we sort the y_i's by their type, T, T-120 or T+120, and the points with the same type are linked to form each individual isophote.

4.3.4 Isophote in Other Interpolation Methods

As mentioned earlier in this chapter, in the general case, obtaining isophote through solving equation (4-3) is not easy. For the family of piecewise polynomial interpolations, equation (4-3) is always a polynomial equation. We have shown how solutions are obtained for the most commonly used bi-linear and bi-cubic methods. Finding isophote in other types of piecewise polynomial interpolations are only briefly discussed as follows.

For quadratic interpolation, its isophote can be found by solving the quadratic equations. Due to its rareness in practical use, the isophote in quadratic interpolation is not derived in this research.

As reviewed in Chapter 2, there are different forms of bi-cubic interpolation. In the previous section, we discussed the case of the most commonly used 4-point Keys' cubic. For the 6-point and 8-point bi-cubic interpolations ([35]), their isophotes can also be obtained by solving the cubic equations as discussed in **4.3.2**. However, their much larger regions of support (6×6 and 8×8, respectively) make solving the equation

extremely lengthy and time consuming. As a result, these types of bi-cubic interpolations are not attempted.

The isophote for piecewise polynomial kernels with orders higher than 3 is also not studied due to both mathematical complexity and lack of practical interest. General polynomial equations higher than the quartic (4th order) are not analytically solvable (the Abel's Impossibility Theorem).

4.3.5 Experimental Results

In order to test the introduced method of finding isophotes in image interpolation, we conduct experiments on several test images. In the experiments, isophotes with different intensity values I_0 are calculated for bi-linear and bi-cubic interpolation, using the methods proposed in sections **4.3.1** and **4.3.2**. We show results obtained from a part of image Camera Man with strong edges, where zigzagging artifacts are most severe. The derived isophotes are shown in Figure 4.7. The test area is also interpolated by a factor of 5, and shown in Figure 4.8 as a comparison of the visual effects in the intensity domain. In Figure 4.7, the spatial resolution of isophote calculation is $\Delta_t = 0.2$. The Catmull-Rom cubic spline (a=-0.5) is chosen as the cubic interpolation function.

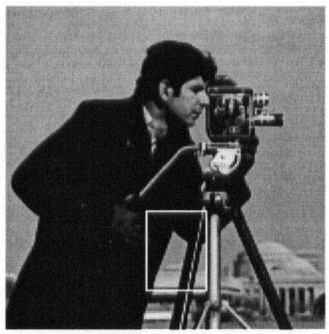
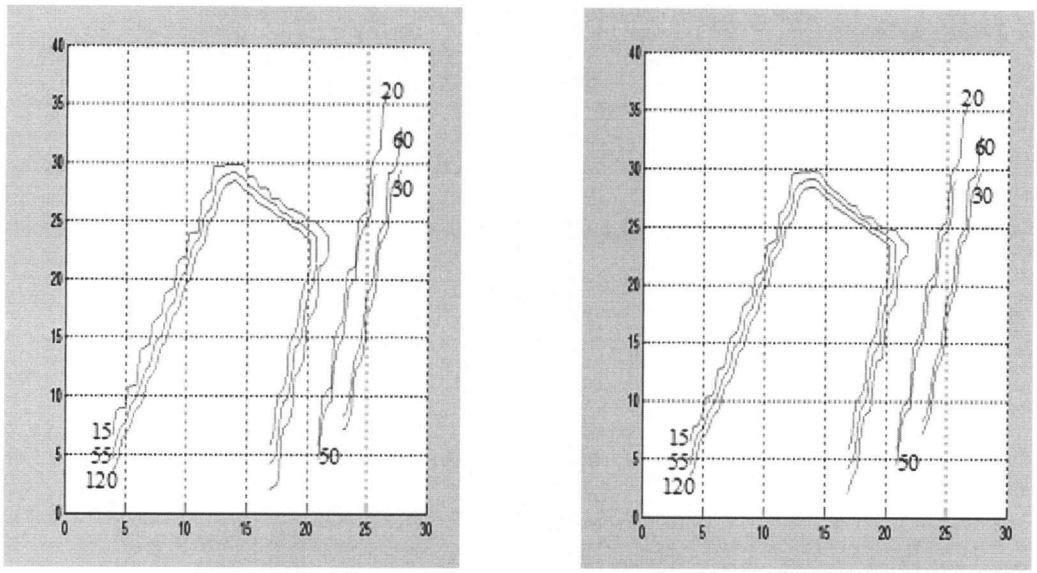
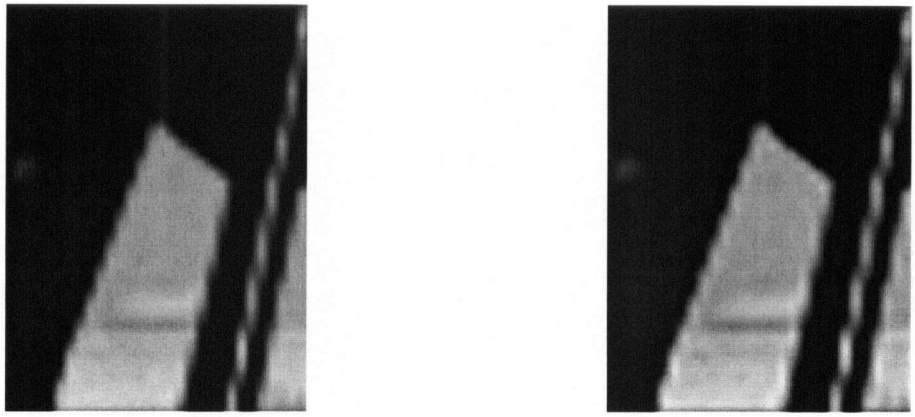


Figure 4.6 Test image Camera Man (enclosed is the test area)



(a) bi-linear (b) bi-cubic

Figure 4.7 Isophotes detected in the test area.



(a) bi-linear (b) bi-cubic

Figure 4.8 Test area interpolated by 5×5.

The test area shown in Figure 4.6 has smooth, mostly straight edges. In this area, the isophotes along the edges are also expected to be smooth and straight. From the test results shown in Figure 4.7, it is clearly observed that, through bi-linear and bi-cubic interpolation, the isophotes become piecewise wavy curves which have unrealistically high curvatures. In the intensity domain (see Figure 4.8), the excessive curvature of isophotes is perceived as zigzagging effect.

4.4 Isophote Analysis for Bi-linear Interpolation

In **4.3.1**, it is shown that the isophotes in the bi-linear interpolation are piecewise hyperbolae (see (4-8)). Because of its mathematical ease to manipulate, we examine the hyperbola isophote of the bi-linear interpolation for further study of zigzagging artifact.

4.4.1 Isophotes and Zigzagging in the Edge Area

Zigzagging artifacts are most visually harmful to image edges. In order to further study the behavior of isophotes in the edge areas, we use the "rotated ideal one-dimension edge" to approximate the real two-dimensional image edges.

An ideal horizontal edge $e=e(x, y)$ has constant values along the horizontal direction, i.e., $e(x, y)$ is reduced to a one-variable function $e(y)$, and has vertically sharp changes (Figure 4.9 (a)). A two-dimensional edge can be approximated by rotating the ideal one-dimensional edge by an angle θ (Figure 4.9 (c)). It is easy to prove that under such a model the isophotes in the edge area are straight lines parallel to the edge's direction (Figure 4.9 (b) and (d)).

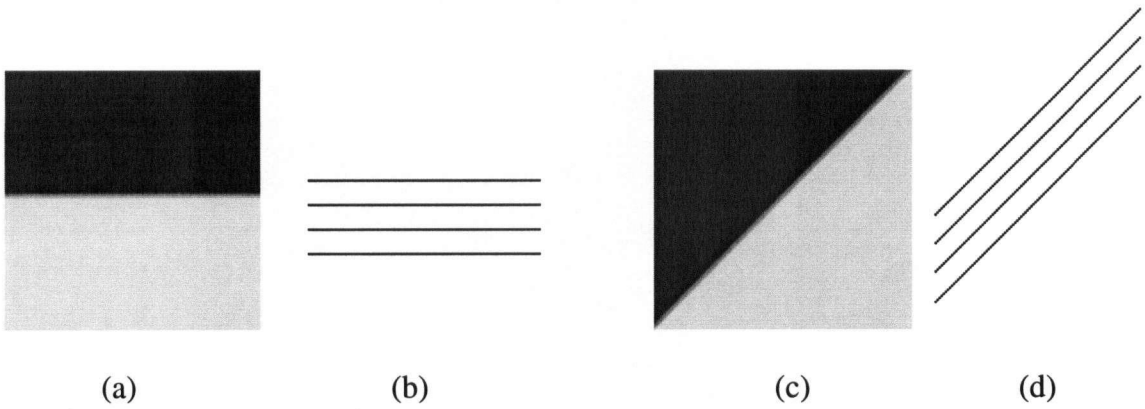


Figure 4.9 The edge and its isophotes.

The edge function in (a) is $e(x, y) = y^3$. (c) is obtained by rotating (a) by 45° counterclockwise. (b) and (d) are illustrative and not exact.

Usually in real-life images, the edges are not precisely one-dimensional, i.e., *e(x, y)* does not precisely reduce to a one-dimensional functions *e(y)*, and neither is the orientation of the edge strictly a constant. However, provided that the sampling rate is sufficiently high, within a small area enclosed by a few pixels, the real edges can be approximated by the rotated one-dimensional edge with sufficient accuracy. Consequently, under this approximation, the isophotes in the edge area can be approximated by near-parallel straight lines illustrated in Figure 4.9 (d).

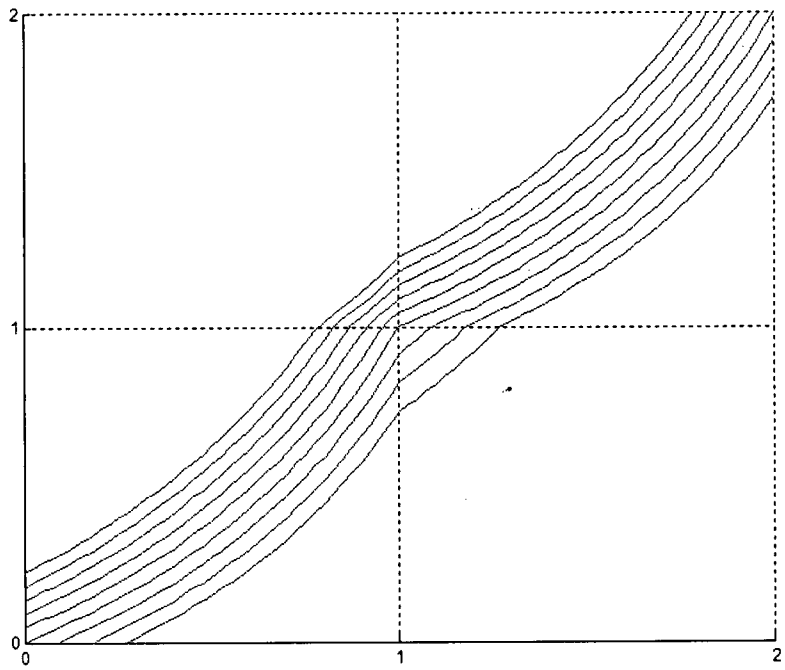


Figure 4.10 Isophotes in the edge area of Figure 4.9(c), reconstructed by bi-linear interpolation.

In the figure, four neighboring interpolation blocks are shown. Each block is enclosed by four parent pixels.

Figure 4.10 shows some of the isophotes in the edge area of the image shown in Figure 4.9 (c), calculated by the method described in **4.3.1**. As discussed in **4.3.1**, in the bi-linear interpolation, the isophotes in each interpolation block have the form of

hyperbolae. We can prove that the isophotes reconstructed by bi-linear interpolation are straight lines only in the following two special cases:

(1) The parent pixels in the edge areas are all located on a plane, or

(2) The original isophotes are horizontal ($\theta=0°$) or vertical ($\theta=90°$).

That is to say, only in the above two special cases, the hyperbolic isophotes reduce to straight lines; for general cases the isophotes are always piecewise hyperbolae, thus zigzagged.

4.4.2 Curvature of Hyperbolic Isophotes

To obtain the curvature of the hyperbolic isophotes, we first re-write hyperbola (4-8) into the following form, which is mathematically more convenient.

$$(x - x_0)(y - y_0) = k \tag{4-20}$$

in which

$$x_0 = -\frac{\beta}{\gamma}, \quad y_0 = -\frac{\alpha}{\gamma}, \quad k = \frac{\gamma(I_0 - A) + \alpha\beta}{\gamma^2} \tag{4-21}$$

For a point (x, y) on a curve described by equation $f(x,y)=0$, its curvature is defined as

$$c(x, y) = \left| \frac{-f_x^2 f_{yy} + 2 f_x f_y f_{xy} - f_y^2 f_{xx}}{(f_x^2 + f_y^2)^{3/2}} \right| \tag{4-22}$$

Particularly for hyperbola (4-20), curvature of a point (x, y) on it is

$$c(x, y) = \frac{2|k|}{[(x - x_0)^2 + k^2 (x - x_0)^{-2}]^{3/2}} \tag{4-23}$$

Further derivation proves that the maximum value of (4-23), i.e., the maximum curvature of hyperbola (4-20), is

$$c_{\max} = \frac{1}{\sqrt{2|k|}} \qquad (4\text{-}24)$$

and c_{max} is obtained at points $(x_0 + \sqrt{|k|}, y_0 + \sqrt{|k|})$ and $(x_0 - \sqrt{|k|}, y_0 - \sqrt{|k|})$. The minimum curvature of hyperbola (4-20) is c_{min}=0, obtained at $x = x_0$ and $x = \pm\infty$, where the hyperbola virtually becomes straight lines when converging to its asymptotes.

For mathematical simplicity, we use the maximum curvature of hyperbola, equation (4-24), to represent the extent of zigzagging within each interpolation block.

From (4-24) it is seen that the greater $|k|$, the less is the curvature. Combining (4-24) and (4-21), it is concluded that the curvature of the hyperbolic isophotes approaches 0 when $\gamma \to 0$. Parameter γ is defined as (refer to **4.3.1** and Figure 4.2)

$$\gamma = A + C - B - D \qquad (4\text{-}25)$$

The value of γ is determined by the values of the parent pixels A, B, C and D, and in the general case $\gamma \neq 0$. However, under the assumption that the isophotes in the edge areas are parallel straight lines, if the interpolation lattice is parallel to the isophotes, parent pixels A and B always have equal values and so do C and D, thus $\gamma = A + C - B - D = 0$ is always met. See Figure 4-11 for illustration.

The above discussion reveals the fact that by making the interpolation lattice parallel to the original isophotes, the reconstructed isophotes will have near zero curvature, regardless of the specific edge function *e(x, y)=e(y)*. And by doing this, zigzagging artifacts in the interpolated image can be eliminated or largely reduced.

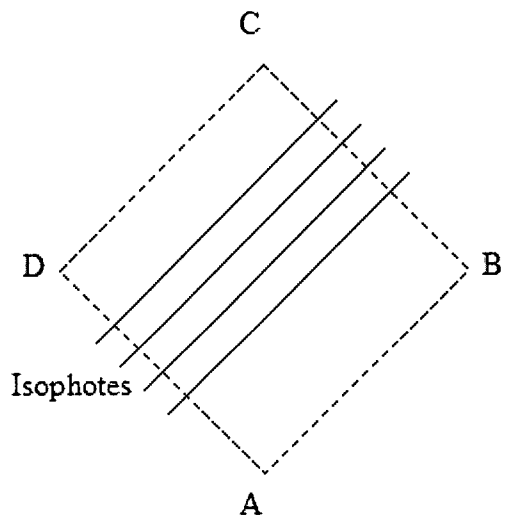


Figure 4.11 Interpolation grid parallel to the isophotes

This conclusion explains the idea held in some recent interpolation methods that the interpolation should be carried along the edges. Also, that the interpolation grid should be parallel to the isophotes is not restricted to edge lines. Instead, this conclusion is applicable to any image areas that have near parallel isophotes, for example, ramps and some textured areas.

Based on the above analysis, in the next chapter, we introduce an interpolation method in which the interpolation grid is adaptively adjusted so it is parallel to the local orientation of isophotes, and by doing this suppresses zigzagging artifacts in the interpolated image.

Chapter 5.
The Directional Interpolation Method

In this chapter, we employ the knowledge gained in Chapter 4 to design an isophote-oriented directional interpolation method. In the proposed method, we generalize the bi-linear interpolation by employing interpolation grid adaptive to the local orientation of isophotes. By doing so, the isophotes of the interpolated image are much smoother than in the traditional methods, thus largely eliminating the zigzagging artifacts. As a result, the subjective quality of the interpolated images is substantially improved.

5.1 The Proposed Directional Interpolation Method

5.1.1 The Parallelogram Interpolation Grid

As discussed in Chapter 4, by employing interpolation grid along the orientation of the isophotes, the resulting isophotes in the interpolated image are piecewise parallel straight lines, and thus the zigzagging effects will be suppressed. Unfortunately, for most of the digital images, the sampling grid is rectangular, i.e., the pixels are always aligned horizontally and vertically, regardless of the orientation of isophotes. As a result, the traditional bi-linear interpolation is always carried using rectangular grids.

For our proposed interpolation method, we design an adaptive grid setting that enables the bi-linear interpolation to be performed along the local direction of isophotes. To do so, we first introduce the parallelogram interpolation grid.

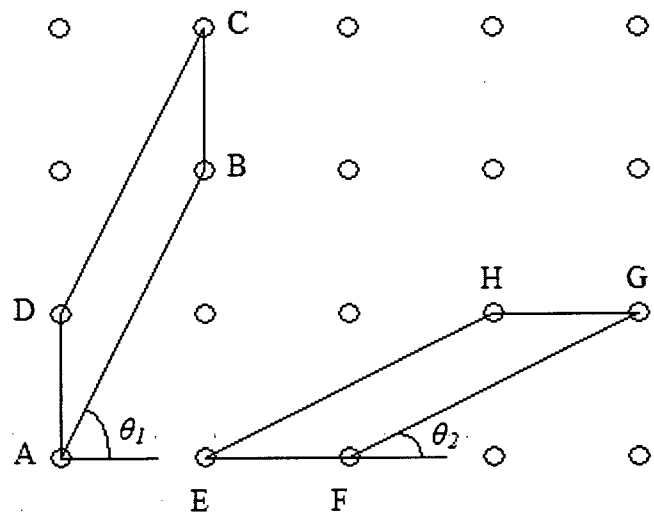


Figure 5.1 The parallelogram interpolation grids.

In the figure, the circles are the grid positions, on which the parent pixels are located.

In Figure 5.1, we show examples of the parallelogram grid, ABCD and EFGH. Parallelogram ABCD has two vertical sides, AD and BC, and two other sides with an angle of $\theta_1 \geq 45^o$, AB and CD. Parallelogram EFGH has two horizontal sides and two sides with an angle of $\theta_2 < 45^o$. We call parallelogram where $\theta_1 \geq 45^o$, such as ABCD, a vertical parallelogram (grid), and parallelogram where $\theta_1 < 45^o$, such as EFGH, a horizontal parallelogram. θ_1 and θ_2 are referred to as the angle of the parallelogram.

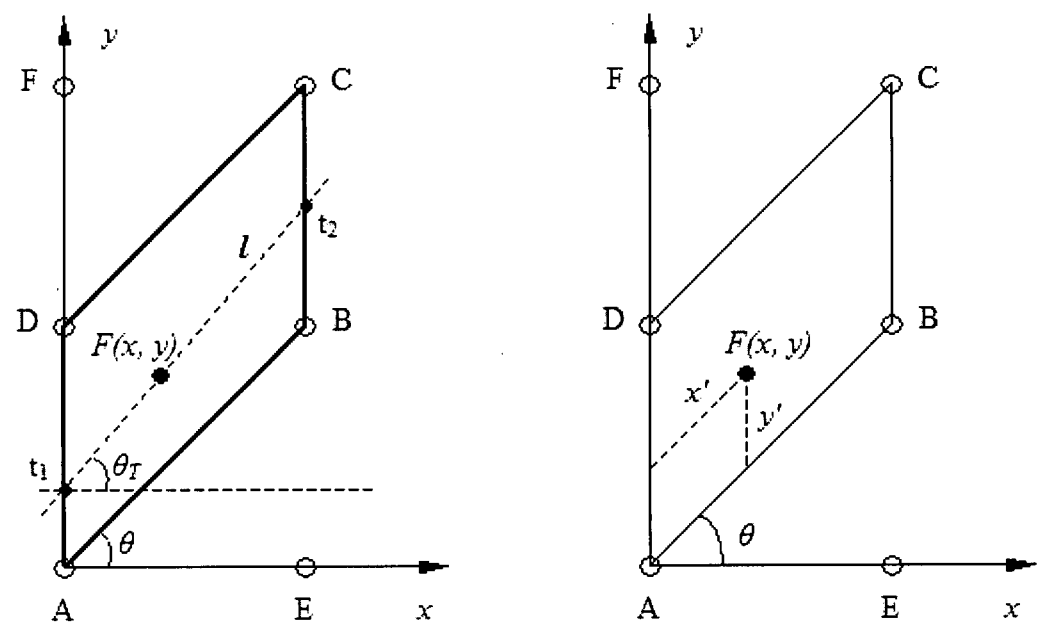
The vertical sides of the vertical parallelogram and the horizontal sides of the horizontal parallelogram always have the length of 1. The angle of the parallelogram, θ, is within the range of 0° to 90°. As can be seen from Figure 5.1, the actual values of θ for

a practical interpolation grid do not form a continuous function, but have the discrete values $\tan^{-1}(n)$ or $\tan^{-1}(\frac{1}{n})$, where n is a positive non-zero integer.

It is easy to prove that the area of any types of parallelograms is always 1, regardless of its angle θ.

5.1.2 The Direction-Adaptive Interpolation

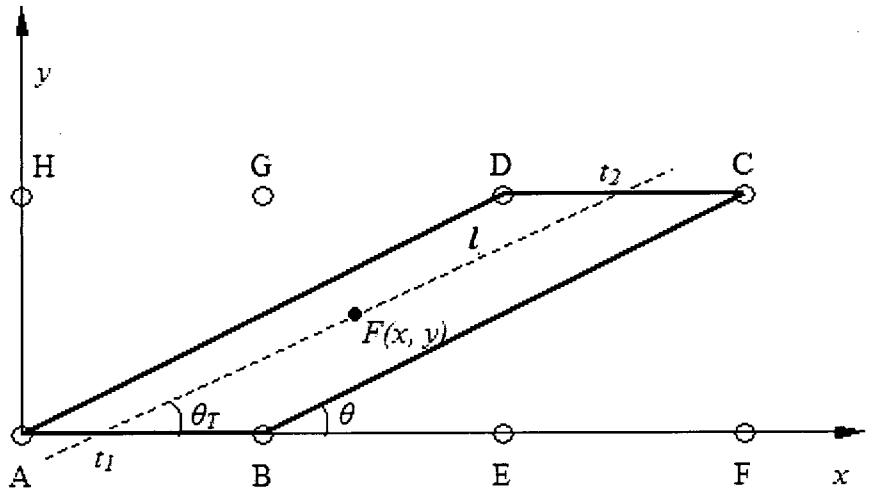
In the proposed interpolation method, a child pixel is always interpolated using intensity values of its parent pixels located on a parallelogram grid that encloses it. We illustrate this process by examples shown in Figure 5.2 and Figure 5.3, where $\theta=45°$.



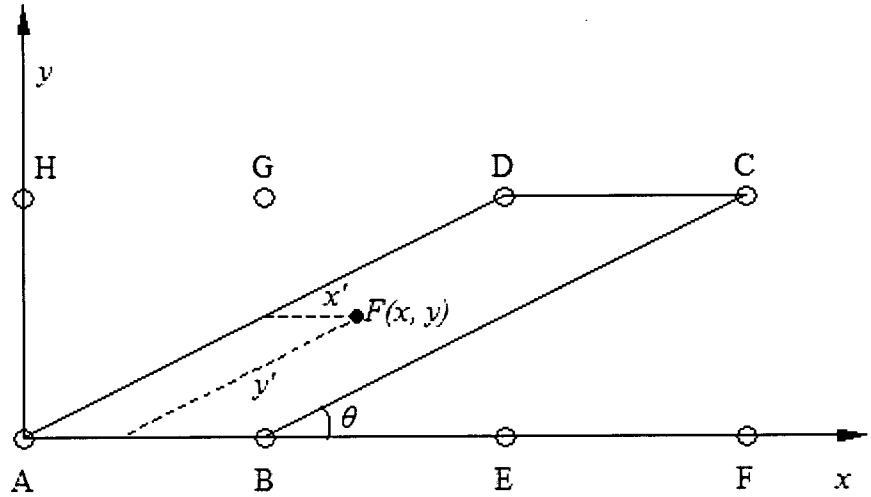
(a) Deciding the parallelogram grid (b) Interpolating the child pixel

Figure 5.2 The vertical parallelogram grid.

Refer to Figure 5.2(a). Assume the value of child pixel $F(x, y)$ is to be interpolated. First, the orientation of the isophote passing through pixel $F(x, y)$ is estimated, yielding θ_T the angle of isophote at (x, y). θ is then obtained by rounding θ_T into the nearest discrete angle of parallelogram. Based on the assumption that, in a small local area, the isophotes are approximately parallel straight lines, a straight line l is drawn passing through point $F(x, y)$ and having the angle θ, the angle of parallelogram.



(a) Deciding the horizontal parallelogram grid.



(b) Interpolating the child pixel.

Figure 5.3 The horizontal parallelogram grid.

In the case of θ_T >45° (illustrated in Figure 5.2), the intersections of line l with the two nearest vertical grid lines, points t_1 and t_2, are found. Grid pixels immediately above and below t_1 and t_2, A, B, C, and D, are chosen as the parent pixels for child pixel $F(x, y)$.

For θ_T <45°, the case of which shown in Figure 5.3, the parallelogram grid is similarly decided. See Figure 5.3(a) for illustration.

If θ_T is approximately 0° or 90°, the isophotes in the traditional interpolation methods are near straight lines and not zigzagged (as discussed in **4.4.1**), thus we employ traditional interpolation using the square grid.

For simplicity, in our discussions we always index the four parent pixels of the interested interpolation grid as A, B, C, and D, starting from the bottom-left vertex and counterclockwise.

After the parent parallelogram is decided for child pixel *F(x, y)*, the value of the child pixel is obtained using (refer to Figure 5.2 (b) and Figure 5.3 (b))

$$F(x, y) = A + \alpha_d x' + \beta_d \ y' + \gamma_d \ x' y' \qquad (5\text{-}1)$$

In (5-1) x' and y' are the re-defined coordinates; α_d, β_d and γ_d are coefficients obtained from parent pixels *A, B, C,* and *D*. Their values are decided as follows.

For the vertical parallelogram,

$$x' = x / \cos\theta, \ \ y' = y - x \cdot \tan\theta \qquad (5\text{-}2a)$$

$$\alpha_d = (B - A)\cos\theta, \quad \beta_d = D - A, \quad \gamma_d = (A + C - B - D)\cos\theta \qquad (5\text{-}2b)$$

For the horizontal parallelogram,

$$x' = x - y \cdot \cot\theta, \quad y' = y / \sin\theta \tag{5-2c}$$

$$\alpha_d = B - A, \quad \beta_d = (D - A)\sin\theta, \quad \gamma_d = (A + C - B - D)\sin\theta \tag{5-2d}$$

Because the proposed interpolation method uses grids that adapt to the direction of isophotes, it is referred to as the ***directional interpolation method***.

5.1.3 Isophotes of the Directional Method

Substituting (5-2) into (5-1) and letting $F(x, y)=I_0$, we can show that for both cases of the horizontal and vertical parallelogram, the isophote of the directional interpolation is given by

$$y = x \cdot \tan\theta + \frac{\gamma_d (I_0 - A) + \alpha_d \beta_d}{\gamma_d^2 (x + \frac{\beta_d}{\gamma_d})} - \frac{\alpha_d}{\gamma_d}, \quad \gamma_d \neq 0 \tag{5-3}$$

Equation (5-3) holds under the restriction that point (x, y) is located within parallelogram ABCD.

Equation (5-3) shows that within each interpolation parallelogram, the isophote of the directional interpolation is composed of two parts: a straight line with angle θ ($y = x \cdot \tan\theta$) and a hyperbola.

Comparing (5-3) with (4-8), it is seen that the traditional bi-linear interpolation can be considered as a special case of the directional interpolation with $\theta = 0$. In the traditional bi-linear method, the interpolation grid is always the rectangle that encloses the child pixels, implying that the angle of the interpolation grid is always assumed to be 0° (or 90°), regardless of the true orientation of the isophotes. In the proposed directional method, the angle of the parallelogram grid θ adapts to the local orientation of the

isophotes. From this point of view, the proposed directional method is a generalized bi-linear interpolation.

The hyperbolic part of the isophote in (5-3) is similar in its form to the hyperbolic isophote of traditional bi-linear interpolation (see (4-8)), but the coefficients α_d, β_d and γ_d are differently defined.

We can prove that if the following condition is met,

(1) In a small image area all isophotes are parallel straight lines with angle θ_T, and

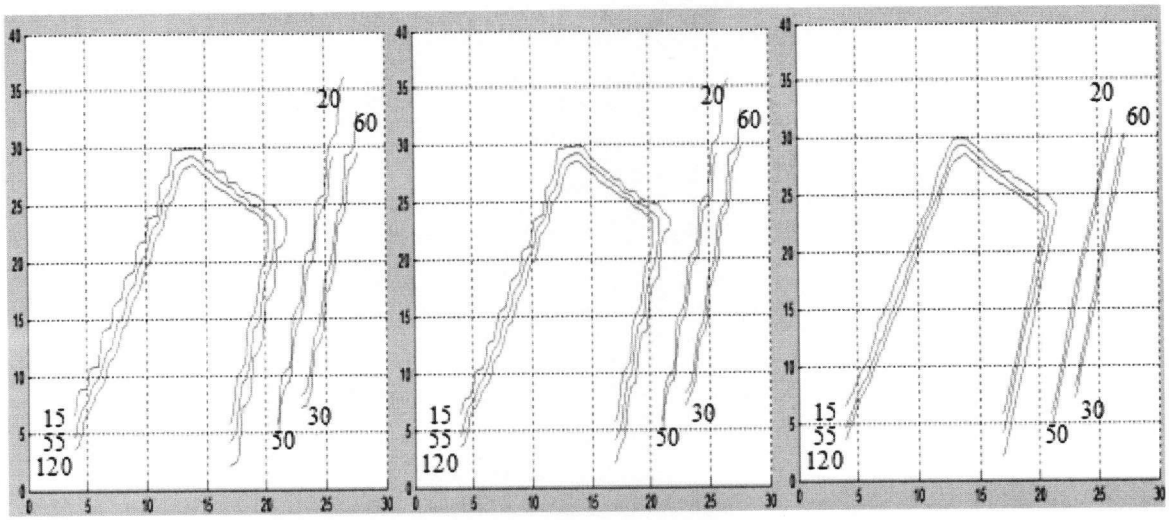
(2) θ_T is accurately estimated, and

(3) the angle of the parallelogram can be precisely chosen, i.e., $\theta = \theta_T$,

(5-3) will reduce to only its linear part. That is to say, under the above conditions, the proposed directional method will generate completely non-zigzagged isophotes.

In practice, it is not likely that the above condition is precisely met, and as a result, the hyperbolic part of isophote (5-3) usually cannot be fully eliminated. However, from our analysis in Chapter 4, when the interpolation grid is parallel (or approximately parallel) to the isophotes, the curvature of the hyperbolic isophote is reduced. Therefore, in our proposed directional interpolation method, the zigzagging in the interpolated isophotes is largely suppressed. As a result, zigzagging artifacts in the interpolated image are effectively suppressed.

In Figure 5.4 we show examples of isophotes interpolated by the directional method. To give a comparison, we also show the isophotes interpolated by bi-linear and bi-cubic method (the same as Figure 4.7 (a) and (b)).



(a) bi-linear (b) bi-cubic (c) the directional method

Figure 5.4 Comparison of isophotes in different interpolation methods

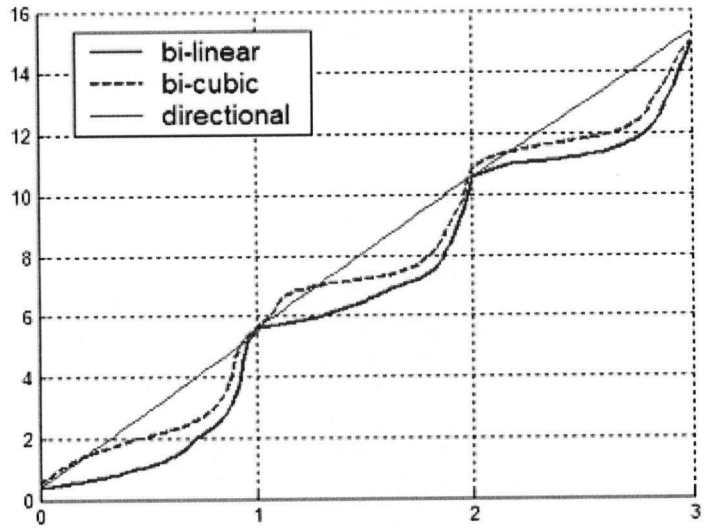


Figure 5.5 A close-up of isophote $I_0=20$ (segment) in Figure 5.4

For a close-up comparison of some zigzagging effects in different methods, we show a segment of isophote $I_0=20$ in Figure 5.5 with a higher resolution.

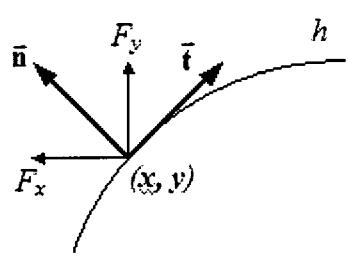
Experiments on other test images yield results similar to those shown above. In section **5.5**, Summaries and Discussions, we calculate the average curvature of isophotes of different interpolation methods, in order to evaluate the performance of the proposed method quantitatively. For detailed results of the curvature analysis, see **5.5**.

The above analysis and results show that the isophotes in the directional interpolation are much smoother than in the traditional methods. Thus we expect the directional method to yield interpolated images with much less zigzagging artifacts. In the following sections, **5.2** and **5.3**, we describe the implementation of the proposed directional method. The images interpolated by the proposed method will be shown in section **5.4**.

5.2 Algorithm of the Directional Interpolation

5.2.1 Estimation of the Isophote Orientation

For each child pixel *(x, y)* to be interpolated, the orientation of the isophote passing through it needs to be estimated. The orientation of an isophote at point *(x, y)* is characterized by the angle of its tangent vector $\vec{\mathbf{t}}$ at that point. See the illustration below.



The angle of $\vec{\mathbf{t}}$ is given by

$$\angle \vec{\mathbf{t}} = \tan^{-1}(-\frac{F_x}{F_y}) \qquad (5\text{-}4)$$

where F_x and F_y are the first partial derivatives of the intensity function *F(x, y)*. By definition of the gradient of *F(x, y)*

$$\nabla F(x, y) = (F_x, F_y),$$

the orientation of the isophote can be obtained by estimating the gradient of *F(x, y)* at point *(x, y)*.

Several gradient estimators are available for estimating the gradient of a continuous image from its sample pixels, such as 2-neighbor (Roberts), the 4-neighbor, the Sobel, the spline-based, and etc. Among them, [2] (page 333) concludes that the improved Sobel operator has the best accuracy in estimating the angle of the gradient.

The improved Sobel operator consists of a horizontal kernel and a vertical kernel, which are

$$S_h = \frac{1}{32}\begin{bmatrix} -3 & 0 & 3 \\ -10 & 0 & 10 \\ -3 & 0 & 3 \end{bmatrix}, \qquad S_v = \frac{1}{32}\begin{bmatrix} 3 & 10 & 3 \\ 0 & 0 & 0 \\ -3 & -10 & -3 \end{bmatrix} \qquad (5\text{-}5)$$

The gradient is estimated by convolving the above kernels with the digital image, i.e.,

$$\hat{F}_x(m,n) = S_h * F(m,n), \qquad \hat{F}_y(m,n) = S_v * F(m,n) \qquad (5\text{-}6)$$

However, the improved Sobel operator cannot be directly used to obtain the gradient of children pixels because it is only applicable to grid pixels. In our proposed method, we use the improved Sobel operator to estimate the gradient of the four nearest parent pixels of a child pixel, and then obtain the gradient of the child pixel by bi-linearly interpolating the parents' gradients, that is (see Figure 5.6 for illustration.),

$$\vec{G}(x,y) = \vec{G}_A + \vec{\alpha}_G \cdot x + \vec{\beta}_G \cdot y + \vec{\gamma}_G \cdot xy \qquad (5\text{-}7)$$

where $\vec{G} = (\hat{F}_x, \hat{F}_y)$ is the estimated gradient for child pixel *F(x, y)*. $\vec{\alpha}_G = (\alpha_x, \alpha_y)$, $\vec{\beta}_G = (\beta_x, \beta_y)$, and $\vec{\gamma}_G = (\gamma_x, \gamma_y)$ are coefficient vectors, which are obtained by

$$\vec{\alpha}_G = \vec{G}_B - \vec{G}_A, \quad \vec{\beta}_G = \vec{G}_D - \vec{G}_A, \quad \vec{\gamma}_G = \vec{G}_A + \vec{G}_C - \vec{G}_B - \vec{G}_D \qquad (5\text{-}8)$$

In (5-8), $\vec{G}_A, \vec{G}_B, \vec{G}_C$ and $\vec{G}_D$ are the gradients of parent pixels A, B, C and D, estimated by the improved Sobel operator.

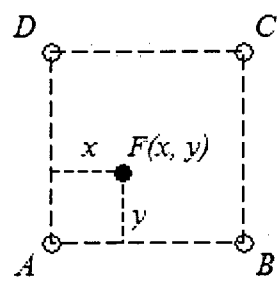


Figure 5.6 Estimating gradient for child pixel *F(x, y)*.

The gradients of parent pixels A, B, C and D are first estimated by the improved Sobel operator. The gradient of pixel *F(x, y)* is bi-linearly interpolated from that of A, B, C and D.

Apart from the proposed gradient estimator ((5-7)), some other operators can also be used to estimate gradient of children pixels. Such examples are the interpolation functions including linear, quadratic, cubic, B-spline, etc. Their derivatives are used to interpolate the gradient of a child pixel with arbitrary position. In our experiment, we tested the proposed gradient estimator and several others, including the ones based on the linear kernel and the Keys' cubic kernel. Most of the results show that our method of gradient estimation is more accurate in estimating the angle of a child pixels' gradient

(and thus that of the isophote), and that the directional interpolation based on the proposed gradient estimator yields better performance.

5.2.2 Interpolating the Children Pixels

At the start of the interpolation procedure, the coordinates of all pixels in the interpolated digital image $F(i, j)$ is calculated by

$$x = i \cdot \Delta_s \ , \quad y = j \cdot \Delta_s$$

in which i and j are the horizontal and vertical indices of the children pixels, respectively; Δ_s is the desired new spatial resolution. Given the interpolation ratio $R \times R$,

$$\Delta_s = \frac{1}{R}$$

For each child pixel, the direction of the isophote passing through it is estimated by the above method proposed in **5.2.1**. Based on the angle information so obtained, the parent grid of each child pixel is decided through the process described in **5.1.2** (see Figure 5.2 (a) and Figure 5.3 (a)). The values of the children pixels are obtained using the directional interpolation described by (5-1) and illustrated by Figure 5.2 (b) and Figure 5.3 (b).

In our method, we set a threshold T_{grad} for the magnitude of gradient of a child pixel. Only those children pixels $F(x, y)$ that satisfy

$$\| \nabla F(x, y) \| > T_{grad} \qquad (5\text{-}9)$$

are interpolated using the directional mechanism described above. Other children pixels are interpolated using the traditional method. Traditional methods, such as the bilinear

and the bi-cubic, can be used to interpolation the non-edge pixels. In our illustration, we use the bi-linear method for simplicity.

The reason of the above thresholding is as follows.

(1) Pixels with low gradient magnitude are likely to belong to smooth areas, which are visually less important. For these pixels, the traditional methods will not generate severe degradations while being computationally economic.

(2) With low gradient magnitude, i.e., small values of F_x and F_y, the angle estimation in (5-4) is sensitive to noise and rounding errors, and therefore less accurate. Guided by inaccurate angle information, the parallelogram grid could be formed by irrelevant parent pixels, leading to large errors in the interpolation.

The process of thresholding the gradient is similar to the gradient-based edge detection ([2]), therefore for simplicity we refer to the image resulting from thresholding as "edge map", and the pixels contained in the edge map as "edge pixels". However, our interpolation method does not require factors such as precise location of the edge, the edges' being thin, or high connectivity of the edges, etc. (Edge-directed interpolation methods often have such requirements. Refer to **3.3.1**). In our algorithm, the magnitude threshold is usually set lower than that used for regular edge detection purposes. As a result, the yielded edge map contains areas, or "clusters", of pixels, rather than very thin edges. Examples of edge maps are shown in Figure 5.7.

5.2.3 Pixels with High Curvatures

In the directional method described above, there is an underlying assumption that within the interpolation grid the isophotes are approximately straight lines. Although this

is true for the majority of situations, in some cases it is not accurate enough. We illustrate one such case in Figure 5.8.

Figure 5.7 Examples of edge maps

In Figure 5.8, the intensity value of a child pixel $F(x, y)$ is to be interpolated. The local direction of the isophote passing through point (x, y) is estimated as that of the straight line $\vec{l}$, for example, and the parallelogram grid is accordingly selected as ABCD. However, in some cases, the isophotes are so highly curved that even within a small

region of few pixels they considerably deflect from the assumed straight line $\vec{l}$. As confirmed in our experiments, such cases usually happen at the sharp corners of image objects, or in areas where multiple image objects overlap, so that abrupt changes of direction occur.

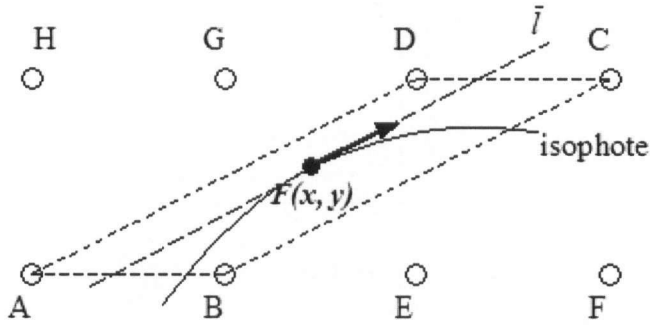


Figure 5.8 A child pixel with high curvature

As shown above, the high curvature of an isophote could result in an improperly selected parallelogram grid, which in turns yields mistakenly interpolated pixel values. These erroneous pixels are visually annoying and must be avoided during the interpolation. Figure 5.9 shows some examples of such mistakenly interpolated pixels.

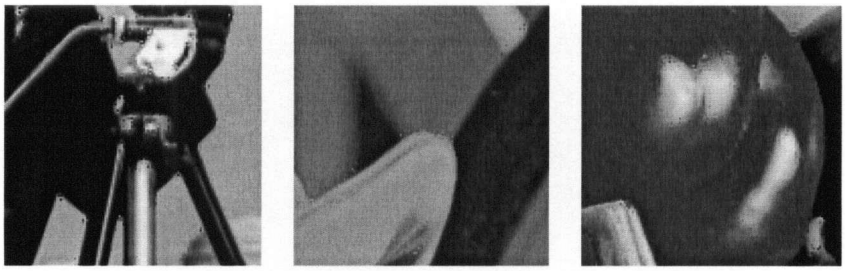


Figure 5.9 Mistakenly interpolated pixels caused by high isophote curvature.

Notice the visually interfering pixels around the corners of some image contours, whose values are obviously irrelevant.

In order to suppress these erroneously interpolated pixels, we introduce a curvature-thresholding mechanism into the directional interpolation. For a child pixel, we estimate the curvature of the isophote passing through it. The directional interpolation is only applied on children pixels with low curvature. For pixels with high curvature, the traditional bi-linear interpolation is used.

For a curve described by equation $F(x, y)=0$, its curvature at point (x, y) is defined as

$$c(x,y) = \left| \frac{-F_x^2 F_{yy} + 2F_x F_y F_{xy} - F_y^2 F_{xx}}{(F_x^2 + F_y^2)^{3/2}} \right| \qquad (5\text{-}10)$$

As in the case of gradient estimation, several operators are available for estimating the curvature. One of the most commonly used curvature estimators is cubic-based. In our experiments, we found that cubic-based curvature estimation is often not accurate enough for our purpose. Instead, we introduce a curvature estimator derived from the gradient estimator described in **5.2.1**.

The definition of curvature can be interpreted as: the differential change of angle of the curve's tangent line divided by the differential length of the curve, over which the change of angle occurs, that is (see Figure 5.10),

$$c(x,y) = \frac{\partial \theta(x,y)}{\partial \vec{l}} \qquad (5\text{-}11)$$

By the properties of the directional derivatives, (5-11) becomes

$$c(x,y) = \frac{\partial \theta(x,y)}{\partial x}\cos\theta + \frac{\partial \theta(x,y)}{\partial y}\sin\theta = \theta_x \cos\theta + \theta_y \sin\theta \qquad (5\text{-}12)$$

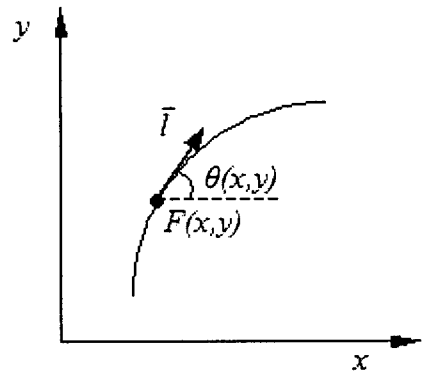


Figure 5.10 Illustration of curvature.

in which $\theta(x, y)$ is the angle of isophote at point (x, y), and

$$\theta(x, y) = \tan^{-1}(-\frac{F_x}{F_y}) \tag{5-13}$$

In order to find θ_x and θ_y of (5-12), let

$$g(x, y) = -\frac{F_x}{F_y} \tag{5-14}$$

Differentiating (5-13) yields

$$\theta_x = \frac{g_x}{1+g^2}, \quad \theta_y = \frac{g_y}{1+g^2} \tag{5-15}$$

Now we resort to our method of gradient estimation described in **5.2.1**, by which F_x and F_y are estimated. Substituting (5-7) and (5-8) into (5-14) and differentiating the result, $g(x, y)$, g_x and g_y in (5-15) are all obtained. The obtained $g(x, y)$, g_x and g_y are in turn substituted into (5-15), yielding

$$\theta_x = \frac{(\alpha_y + \gamma_y \cdot y) \cdot F_x - (\alpha_x + \gamma_x \cdot y) \cdot F_y}{F_x^2 + F_y^2}, \quad \text{and}$$

$$\theta_y = \frac{(\beta_y + \gamma_y \cdot x) \cdot F_x - (\beta_x + \gamma_x \cdot x) \cdot F_y}{F_x^2 + F_y^2} \tag{5-16}$$

Eventually, we substitute (5-16) and (5-13) into (5-12), and the estimated curvature *c(x, y)* is calculated.

With the curvature of the children pixels estimated, a threshold T_{curv} is set. Children pixels *F(x, y)* that have high curvatures, i.e.,

$$|c(x, y)| > T_{curv} \tag{5-17}$$

are NOT interpolated by the directional grids. Instead, their values are obtained through the traditional bi-linear method.

In our algorithm, the threshold T_{curv} is made adaptive to θ, the angle of the parallelogram. Refer to Figure 5.8. The closer θ is to 0° or 90°, the thinner the parallelogram, and thus the less curved the isophote needs to be, if it is to be contained by the parallelogram grid. Therefore, in our algorithm, we set T_{curv} as

$$T_{curv} = \frac{t}{\tan\theta}, \text{ for a vertical parallelogram; } T_{curv} = \frac{t}{\cot\theta}, \text{ for a horizontal parallelogram.} \tag{5-18}$$

In (5-18) t is a constant. In our experiments we find t= *0.8~1.5* yields satisfying results in most situations.

Although the estimation of the curvature is a fairly resource consuming operation, in our method it is only carried on the edge pixels, which only occupy a minority (usually less than 20%) of the total pixels. The non-edge pixels are already filtered out in the first stage of the interpolation algorithm by the gradient thresholding, and thus will not have their curvature estimated. Also, by using the proposed method of curvature estimation,

most information needed for calculating the curvature is already obtained when estimating the gradient, and thus can be reused. As a result of the above factors, the curvature estimation is not particularly time consuming in our algorithm.

The experimental results show that the curvature thresholding effectively suppresses pixels with high curvatures that are inappropriately interpolated. As a result, the visual satisfaction of the interpolated images is considerably improved.

In the following section, we continue to improve the proposed directional method by discussing another type of visually important image features, the ridges. Before that, we here show in Figure 5.11 and Figure 5.12 some images interpolated by the directional method so far proposed.

Figure 5.11 Camera Man interpolated by 2×2 using the proposed method

Figure 5.12 Lena sub-sampled (upper) and expanded (lower) by the proposed method

5.3 Interpolation of the Ridge Areas

5.3.1 Ridgelines and Artifacts

In the directional interpolation method proposed in sections **5.1** and **5.2**, the orientation of the isophotes is estimated by the gradient operator applied on the original image. In our further study, we find that although the gradient-based orientation estimation is effective on image features such as edges and ramps, it does not work equally well for some image features with low gradient magnitude. One type of those image features is the "ridgelines", or "ridges".

Our formal definition of a ridge is given later in **5.3.2**. The most commonly seen examples of the ridges are bright (or dark) thin lines. See Figure 5.13 for two examples.

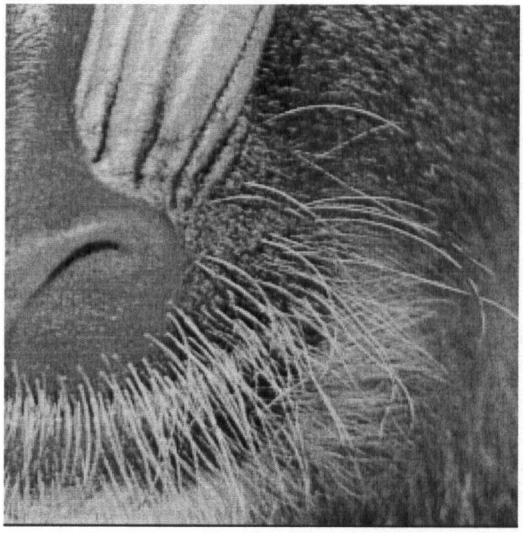


Figure 5.13 Examples of ridgelines.

The hair (Mandrill) and the bright lines inside the legs of the tripod (Camera Man) are typical ridgelines

A closer study of the ridges reveals that the ridges have several types, not all appearing in the form of thin lines. We show three different types of ridges in Figure 5.14 by illustratively drawing the profiles across the ridgeline.

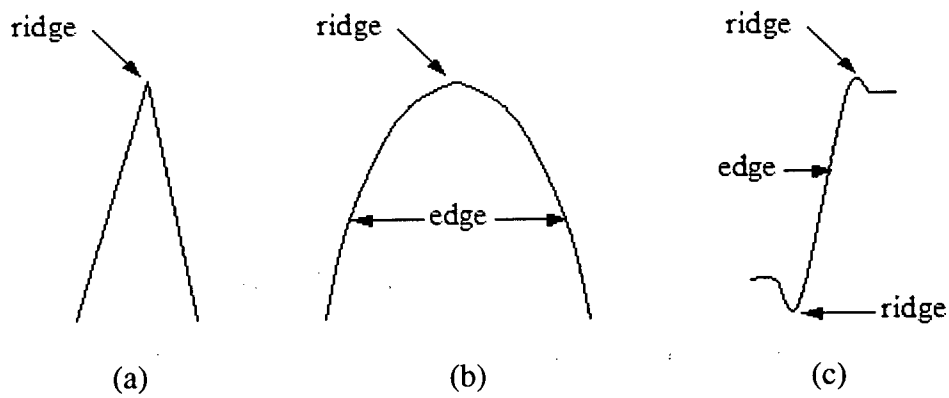


Figure 5.14 Cross section of different types of ridges

Figure 5.14 (a) and (b) both correspond to the cross section of an image object that has a ridgeline near its top and slopes on both sides. In (b) the ridgeline is in the middle of the two edges. The case of (a) corresponds to a very thin line. In (c), the ridges occur along an edge in the form of overshootings (ringing). Overshooting can be caused by limited bandwidth of the original acquisition system.

All three cases of ridges suffer artifacts after interpolation. As an example, in our experimental results shown in Figure 5.11, notice the artifacts along the bright lines in the legs of the tripod and the handle. This phenomenon is also observed in other results not shown here. Degraded by artifacts, the ridgelines appear twisted, zigzagged, and in some cases as broken lines. All traditional interpolation methods reviewed in Chapter 2 suffer from these ridge-related artifacts.

Our observations also show that although the ridges of cases (b) and (c) are usually less visible than case (a), both case (b) and (c) are very common in images even though they are usually not as noticeable. These ridge-related artifacts often degrade the visual appearance of the associated edges after interpolation. Such ridge-related artifacts can be observed along the rim of the hat in Figure 5.12, and some of the edges in Figure 5.11.

The proposed directional interpolation method has proved to be very effective in removing the zigzagging artifacts in edge areas. However, the artifacts associated with ridgelines are not solved yet. Next, we study how to improve the proposed directional method in ridge areas.

5.3.2 Isophote Model of Ridges

A typical ridgeline often appears to have a constant intensity value over its stretch, and if so, the isophotes along this ridgeline are all parallel straight lines, in which case directional interpolation should yield good results. However, a close examination of the ridges reveals that the intensity values along a ridgeline still vary, although usually slowly. As a result, the isophotes along the ridgeline are nested closed curves, instead of straight lines. Figure 5.15 illustrates one case of ridgelines. In Figure 5.15, the intensity value of the ridgeline is the highest at the bottom-right corner, decreases toward the center, and increases toward the top-left corner after passing through the center point. Most commonly, the intensity of the ridge fluctuates along it, and in such a case the model could be drawn as multiple sets of the nested curves shown in Figure 5.15 aligned along the ridge direction, concatenating each other.

(a) The isophotes (b) The intensity

Figure 5.15 Isophotes in the ridge area.

For better visibility, the isophotes close to the center are not shown.

Referring to Figure 5.15, we use homocentric ellipses to approximate the isophotes in the ridge area. One reason of the ellipse approximation is its mathematical simplicity. Under this model, we define the **ridge point** as ***the point on each ellipse isophote that has the maximum curvature***. Accordingly, the ridgeline is defined as ***the straight line that links all the ridge points***.

As stated earlier in this section, ridge points have low gradient magnitudes. This can be briefly explained by examining Figure 5.14. As can be seen from Figure 5.14, the ridge point is near the peak (or valley) of the ridge's cross section, and therefore has a near-0 partial derivative along this direction. Assuming that the intensity value along the ridge direction is changing slowly, the partial derivative along the ridge direction is also small. The gradient, formed by the partial derivatives along and across the ridgeline, thus has a low magnitude. This will be mathematically proven in the next sub-section when we prove that the ridge pixel is very close to either the horizontal maximum or vertical maximum, or both.

The low gradient magnitudes for the ridge pixels impose the following two major difficulties on the mechanism of the proposed direction interpolation.

(1) With $\| \nabla F(x, y) \| \rightarrow 0$, the ridge pixels are likely to be filtered out by the gradient thresholding at the early stage of the interpolation.

(2) With F_x and F_y having small values, $\frac{F_x}{F_y}$ is close to type $\frac{0}{0}$. As a result, the direction of the isophote, $\tan^{-1}(-\frac{F_x}{F_y})$, is very sensitive to noise and errors (typical errors are those introduced by the gradient estimation and the rounding error when digitizing the original image).

Note that (2) is also the reason why the orientation cannot be accurately estimated for pixels in the smooth areas, and consequently they are excluded from the proposed directional interpolation by the gradient thresholding (see **5.2.2**).

With the presence of the above two problems, the directional interpolation does not effectively detect the ridge areas, nor adapts the interpolation grid appropriately to the correct direction for ridge pixels. In fact, the above two problems also affect all gradient-based edge-detection (including Sobel, Canny, Spline-based, and so on), orientation estimation, and the interpolation methods so based.

Apart from the above two problems associated with ridge pixels in general, there is another difficulty specific to the proposed directional method. As illustrated in Figure 5.15, on the ridgeline, the isophotes are no longer near-parallel straight lines. Instead, they are highly curved, and it is proven that on the ridgeline, the isophotes are perpendicular to the orientation of the ridgeline, instead of parallel to it. That is to say,

even if the ridge pixels were detected and the orientation of the isophotes properly estimated, the interpolation parallelogram would be set with a totally wrong direction.

Extensive studies and rigorous mathematical derivations of how interpolation should be carried in the ridge areas in order to reduce the zigzagging artifacts are much more difficult than for the edge areas, and such analysis will not be conducted with full strength within the scope of this research. However, we here adopt the reasonable idea that in the ridge areas the interpolation should be carried along the ridgeline, rather than the along the isophotes as in the case of edge areas. In the next sub-section, we design a scheme through which the orientation of the ridgelines is estimated and the corresponding directional interpolation is carried.

5.3.3 Interpolating the Ridgelines

In order to estimate the orientation of ridgelines, we first examine the ellipse-based isophote model of the ridge area introduced in **5.3.2**.

Assume that an ellipse has a major axis a and a minor axis b, and its major axis is at an angle θ with the x-axis. The parametric form of such an ellipse is

$$\begin{cases} x = a\cos\varphi\cos\theta - b\sin\varphi\sin\theta = \omega(\varphi) \\ y = a\cos\varphi\sin\theta + b\sin\varphi\cos\theta = \eta(\varphi) \end{cases} \tag{5-19}$$

in which φ=0~2π is the angle parameter.

From (5-19) it follows that

$$\frac{dy}{dx} = \frac{\eta'(\varphi)}{\omega'(\varphi)} = \frac{-a\sin\varphi\sin\theta + b\cos\varphi\cos\theta}{-a\sin\varphi\cos\theta - b\cos\varphi\sin\theta} \tag{5-20}$$

Let $\eta'(\varphi)=0$ and $\omega'(\varphi)=0$, i.e., $\frac{dy}{dx}=0$ and $\frac{dy}{dx}=\infty$, respectively, and denote the yielding φ as φ_h and φ_v, respectively. Derivations show that φ_h and φ_v satisfy

$$\varphi_h = \tan^{-1}(\frac{b}{a}\cot\theta) \tag{5-21a}$$

$$\varphi_v = -\tan^{-1}(\frac{b}{a}\tan\theta) \tag{5-21b}$$

Meanwhile, because the ellipse is an isophote, it must satisfy

$$\frac{dy}{dx} = -\frac{F_x}{F_y} \tag{5-22}$$

Equation (5-22) implies that φ_h and φ_v correspond to points on which $F_x=0$ and $F_y=0$. Therefore, non-strictly but without losing practicality, it is concluded that φ_h and φ_v correspond to the horizontal maximum and vertical maximum of function $F(x, y)$ (here we refer to both maximum and minimum as maximum, when the meaning is clear from the context).

In our model of isophotes in the ridge areas, the ellipses are usually very thin, i.e., $b<<a$. With $b<<a$, (5-21a) reveals that when $\theta>45°$ (so that $cot\theta < 1$), $\varphi_h \rightarrow 0$; similarly, when $\theta<45°$, there is $\varphi_v \rightarrow 0$.

It is easy to prove that the ridge point, defined as the point that has the greatest value of curvature on the ellipse, is the intersection of the major axis with the ellipse (here we only examine one of the two such points), which corresponds to $\varphi = 0$.

Because φ_h and φ_v always have different signs (see (5-21)), $\varphi = 0$ is always between φ_h and φ_v. Combining this fact with the conclusion drawn above, it is concluded that when $\theta>45°$, the ridge point is very close to the horizontal maximum; when $\theta<45°$, the ridge point is very close to the vertical maximum. Figure 5.16 illustrates the case of

$\theta>45°$. In this case the ridgeline is very close in both angle and location to the line that links the horizontal maxima, and is away from the line that links the vertical maxima.

Based on the above analysis, we detect the lines of horizontal maxima and vertical maxima, and use them to approximate ridges with $\theta>45°$ and $\theta<45°$, respectively. The case of searching a ridgeline with $\theta<45°$ is illustrated in Figure 5.17.

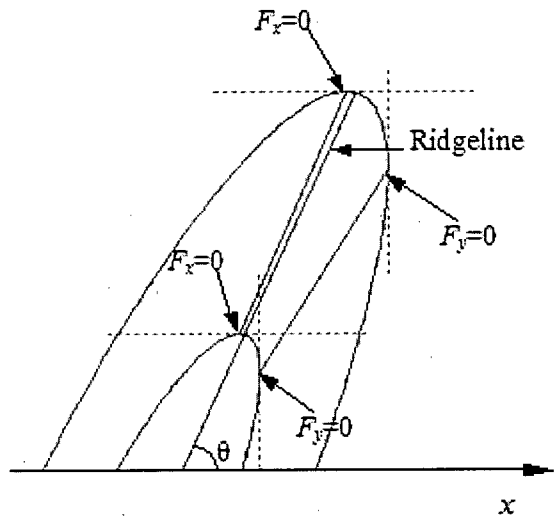


Figure 5.16 Ridgeline and the lines of maxima.

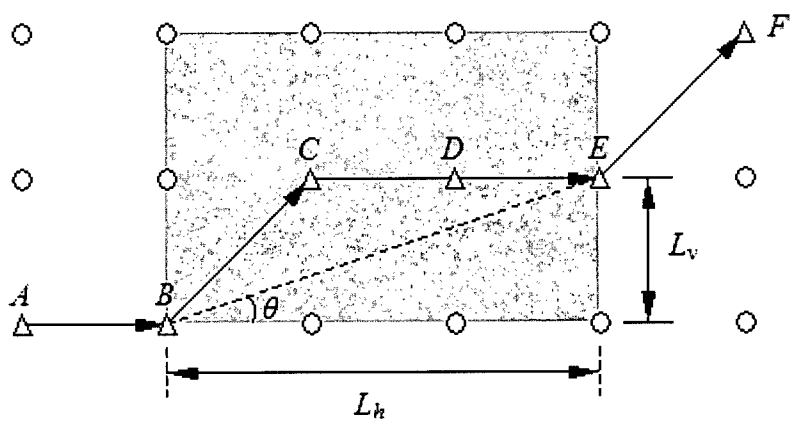


Figure 5.17 Tracking a ridgeline with θ<45°.

In the figure, triangles are vertical maximum pixels, and circles are other grid pixels.

Refer to Figure 5.17. In the original image, we search for ***runs of vertical maxima*** — consecutive vertical maximum pixels aligned horizontally — for each row. For example, A-B and C-D-E are two runs of vertical maxima. A ridgeline starts with a single run of maxima, say run A-B. If another run of vertical maxima is found in the neighboring row (upper or lower) and its starting pixel diagonally neighbors the ending pixel of current ridgeline (pixel B), the new run is connected to the ridgeline and the ridgeline grows. In this case, run C-D-E is connected because pixel C neighbors pixel B diagonally. The ridgeline grows until no more runs can be connected to it.

In our algorithm, only ridgelines with 3 or more runs connected are considered valid, otherwise they are considered too short to be interested, or likely to be interference, and are ignored. In a ridgeline, each run of maxima is a segment, and the orientation of the ridgeline for each segment is decided by dividing its vertical and horizontal span, i.e., $\tan\theta = \frac{L_v}{L_h}$, as illustrated in Figure 5.17. For the case of $\theta<45°$, there is always $L_h=1$. Different segments of the ridgeline can have different θ angles because L_h's for different segments are different.

In the case of $\theta>45°$, the ridgeline is detected by searching for runs of horizontal maxima. The detailed process is similar to the case of $\theta<45°$ and is not separately explained. In the case of $\theta\approx45°$, the maxima most often appear both horizontal and vertical (two-dimensional maxima), and in this case the ridgeline is detected by both horizontal searching and vertical searching.

The ridgelines formed by minimum pixels, or the valleys, are also detected by the same mechanism as described above, but with maxima replaced by minima. Similar to the case of edges, ridgelines that are almost horizontal or vertical usually suffer less from

artifacts. Therefore we neglect such ridges and use traditional bi-linear method to interpolate them. Figure 5.18 shows some examples of detected ridges.

When the orientation of each segment of the ridgelines is decided, children pixels located within 1-pixel distance from the ridgeline (e.g., the shaded area in Figure 5.17 for segment C-D-E) are interpolated using the directional method described earlier in this section, with θ the angle of the parallelogram.

Summarizing **5.1** through **5.3**, the completed directional interpolation method consists of the following 5 major steps.

(1) Gradient thresholding, through which the edge pixels are identified.

(2) For pixels identified in step (1), their orientation is estimated. During this step, the curvature thresholding is performed. Those pixels with high curvatures are excluded from orientation estimation.

(3) Detect ridgelines on all pixels, and estimate the orientation of ridgelines.

(4) For all pixels whose orientation is obtained through step (2) and (3), their values are interpolated by the directional interpolation method.

(5) For all pixels excluded from step (4), their values are obtained using bi-linear interpolation.

Figure 5.18 Examples of detected ridges.

In the figures, the original image is set as background with lowered intensity level. The ridgelines are highlighted. Note that ridgelines that are almost horizontal or vertical are not shown.

5.4 Experimental Results

With the proposed directional interpolation method completed, we conduct extensive experiments in order to test the performance of the proposed method. In this section, we show some of the experimental results.

In our experiments, the original test images are sub-sampled and then expanded, and the expanded images have the same size as the original ones. The expansion ratio in each direction is chosen as $R=2$ unless otherwise specified, and correspondingly the down-sampling ratio is $1/R$. Some of the images, such as the Camera Man, are originally obtained with low resolutions. In such cases they are not sub-sampled and only expanded by $R=2$. To compare the proposed method to the traditional methods, we also show the result obtained through the bi-cubic interpolation, using the Catmull-Rom cubic spline.

The results of image Camera Man, Lena, Zelda, Pepper, Tropic and Mandrill are shown in Figures 5.19 - 5.24. For image Camera Man and Tropic, results from replication, bi-linear, and quadratic interpolation (which are not shown in this section) can be seen in Figure 2.6 and 2.7. For some of the images, we also show the results from the directional method <u>with</u> and <u>without</u> ridge interpolation in order to demonstrate its effect. If not specified, the directional method is <u>with ridge interpolation</u>.

For all images shown below, the size after expansion is 512×512 except Tropic (320×320), and the spatial resolution of display is 96 DPI.

Figure 5.19 (a) Expanded by bi-cubic interpolation.

Figure 5.19 (b) Expanded by directional interpolation.

See the bright lines on the legs of the tripod for ridges. Compare with Figure 5.11, which does not have the ridges specially interpolated

Figure 5.20 (a) Expanded by bi-cubic interpolation.

Figure 5.20 (b) Expanded by directional interpolation.

See the rim of the hat and the hair area for differences between with and without ridge interpolation (Figure 5.12).

Figure 5.21 (a) Expanded by bi-cubic interpolation.

Note: Expansion Ratio $R=2.5$.

Figure 5.21 (b) Expanded by directional interpolation.

Expansion Ratio $R=2.5$.

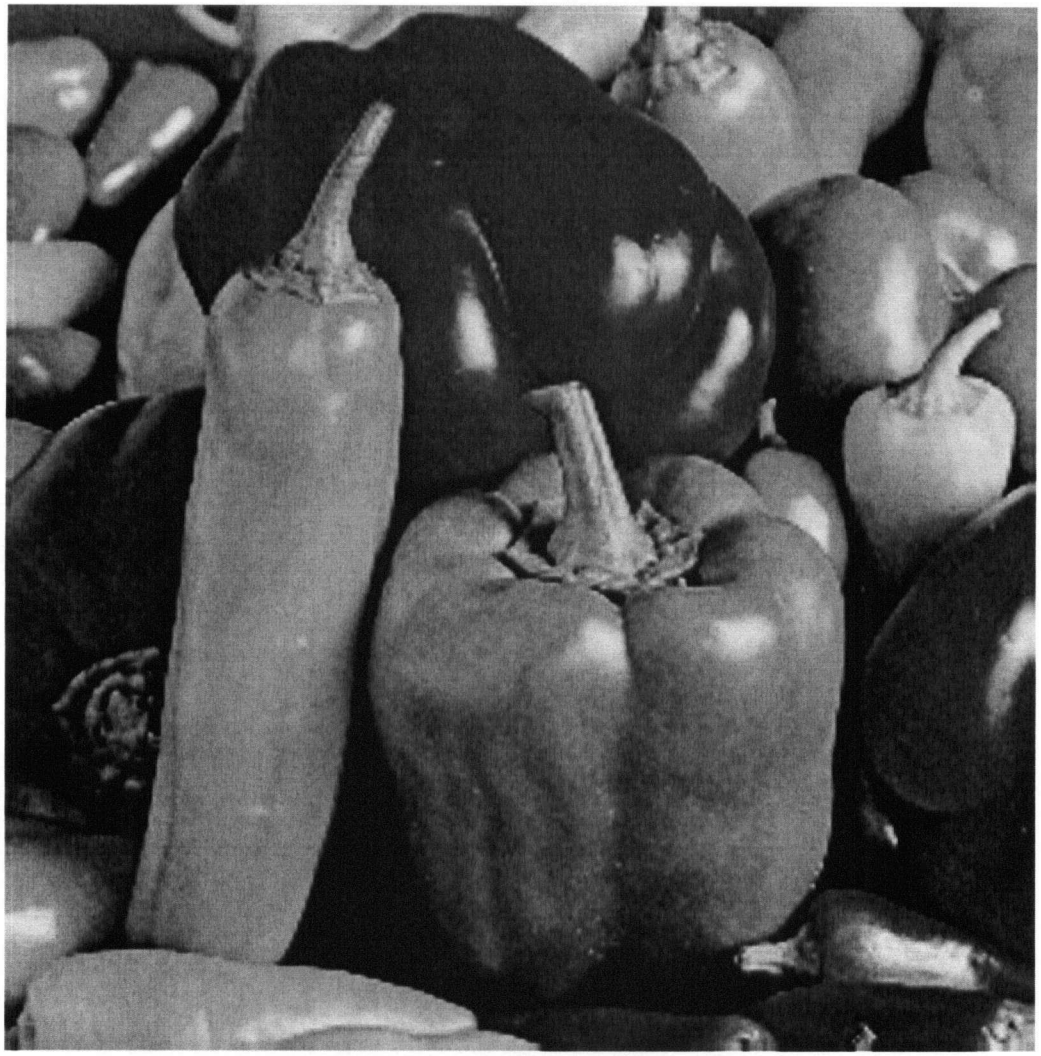


Figure 5.22 (a) Expanded by bi-cubic interpolation.

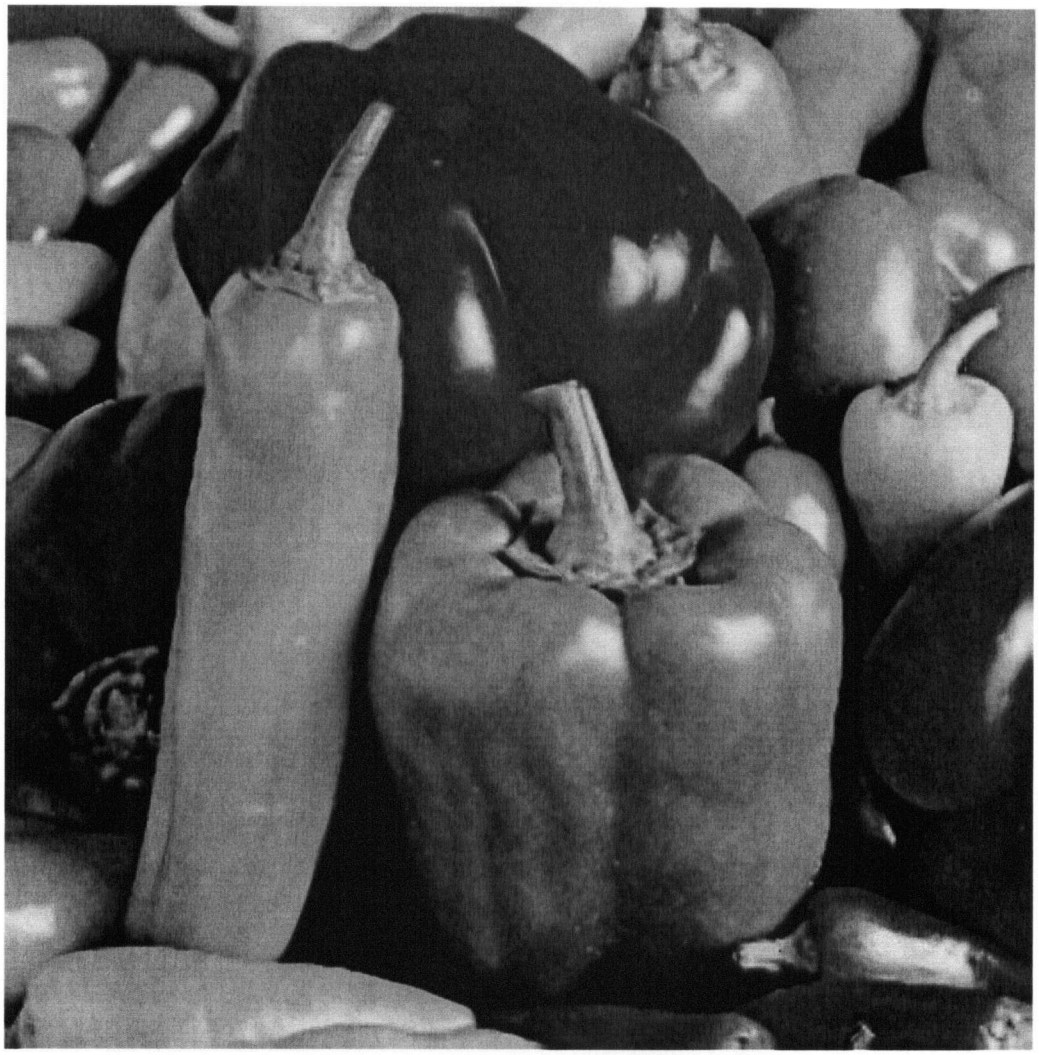


Figure 5.22 (b) Expanded by directional interpolation

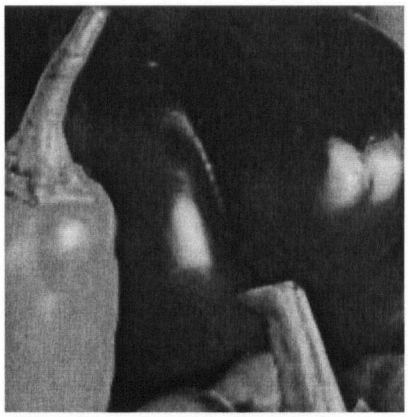
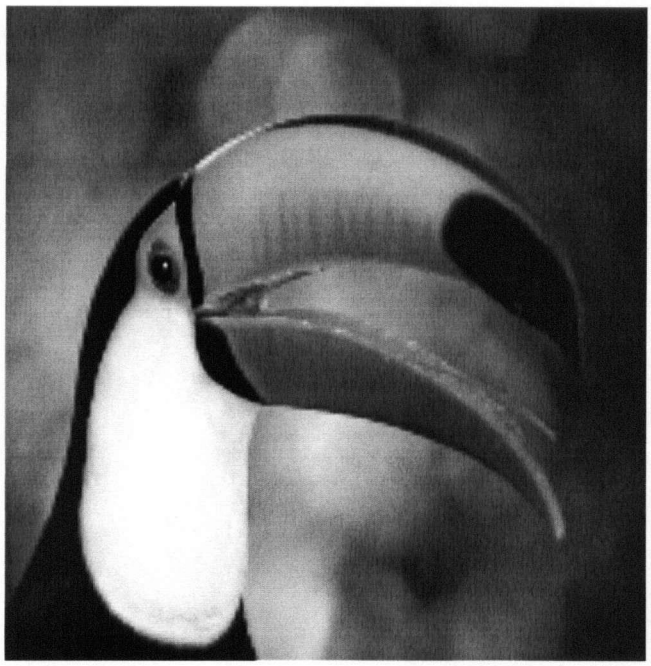
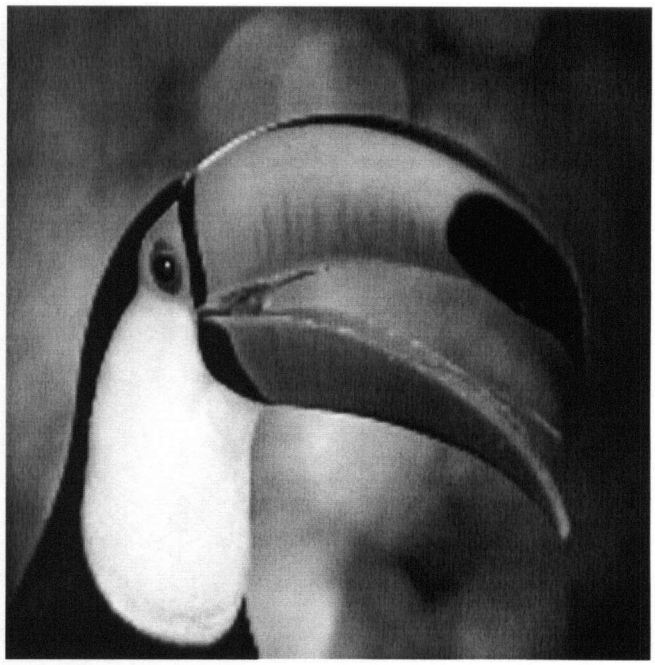


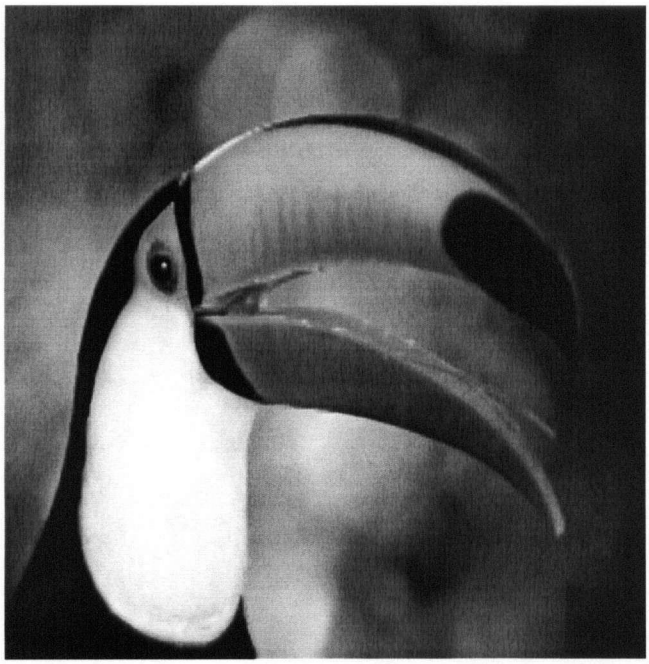
Figure 5.22 (c) Expanded (part) by the directional method <u>without ridge interpolation</u>.



(a) Expanded by bi-linear interpolation



(b) Expanded by bi-cubic interpolation



(c) Expanded by directional interpolation

Figure 5.23 Tropic expanded.

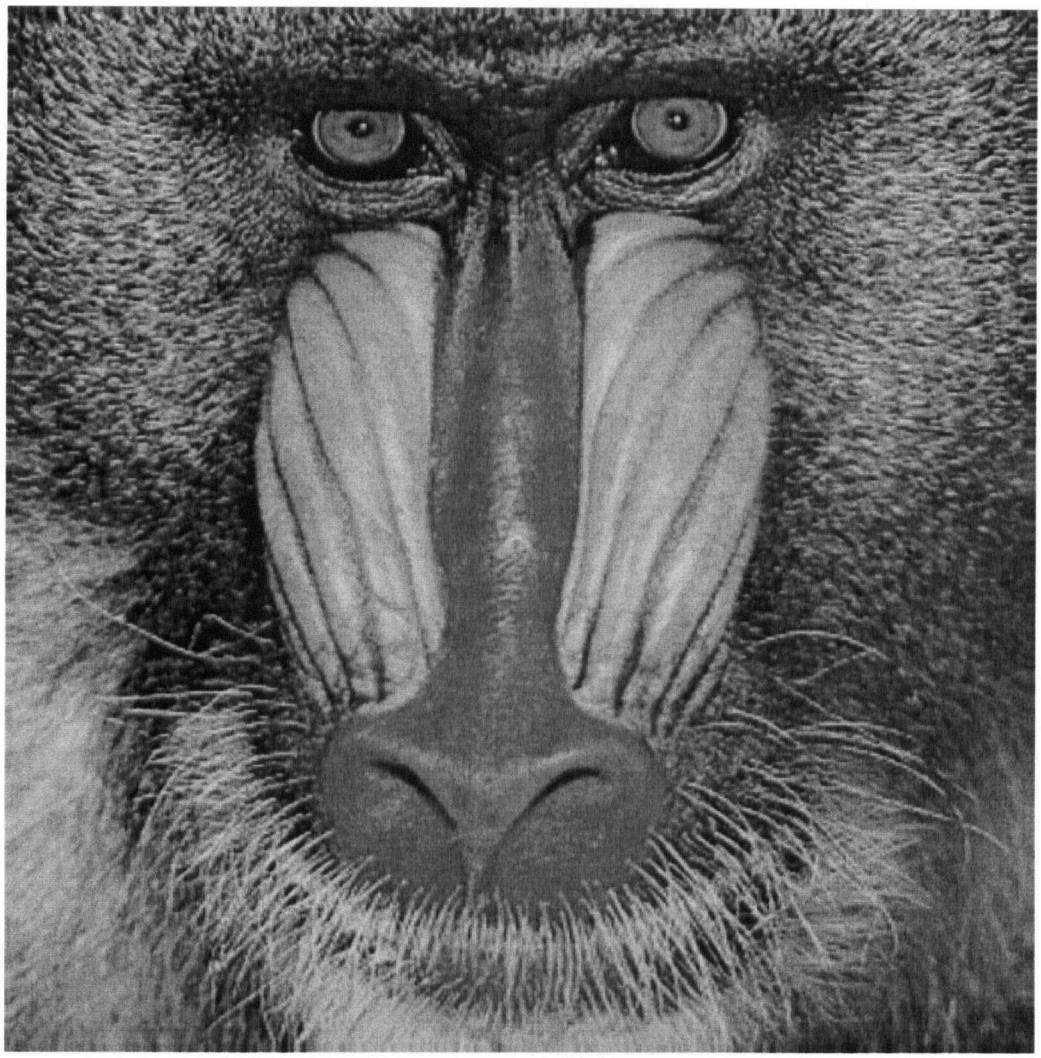


Figure 5.24 (a) Expanded by bi-cubic interpolation

Expansion ratio $R=1.5$.

Notice the artifacts in the hair areas.

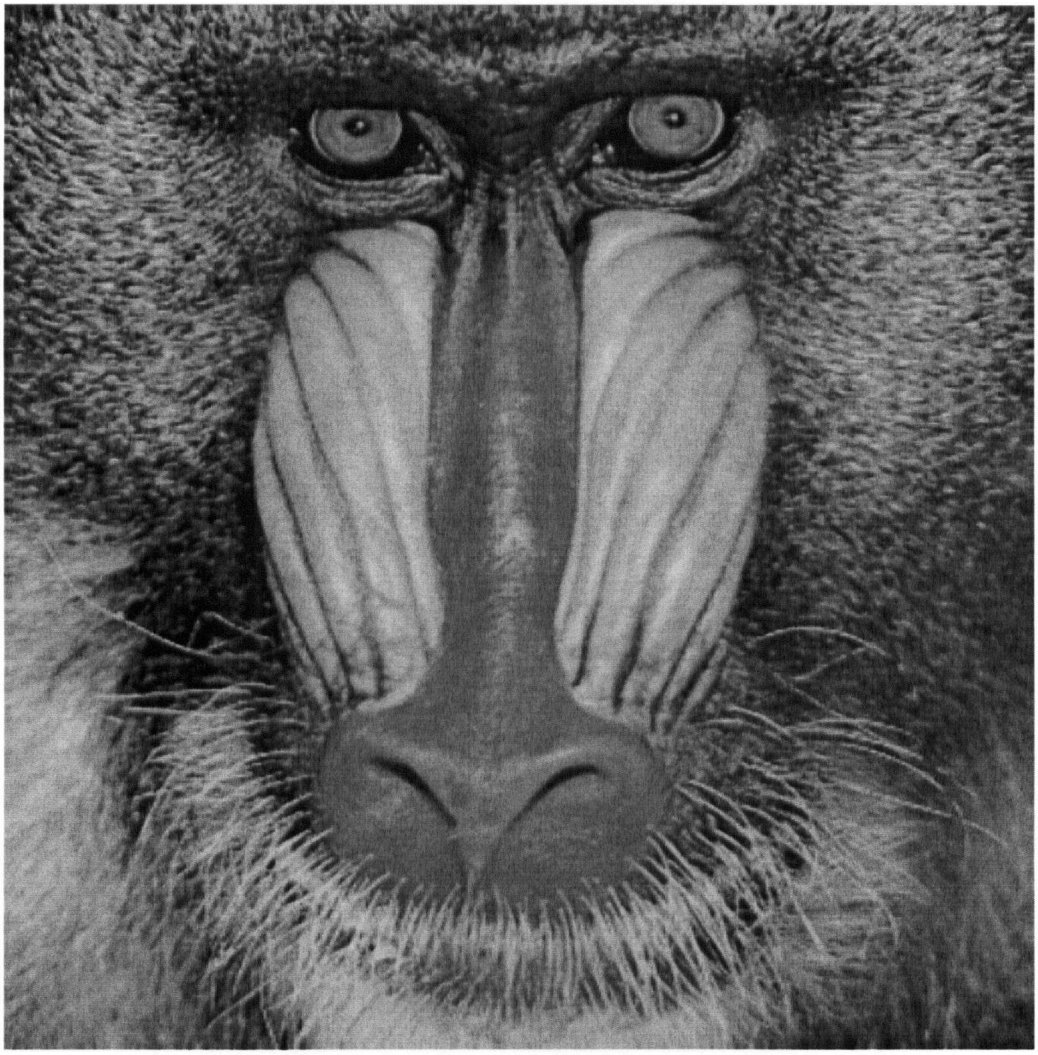


Figure 5.24 (b) Expanded by the directional method <u>without ridge interpolation</u>

Expansion ratio $R=1.5$. Notice that the hair areas still have some artifacts.

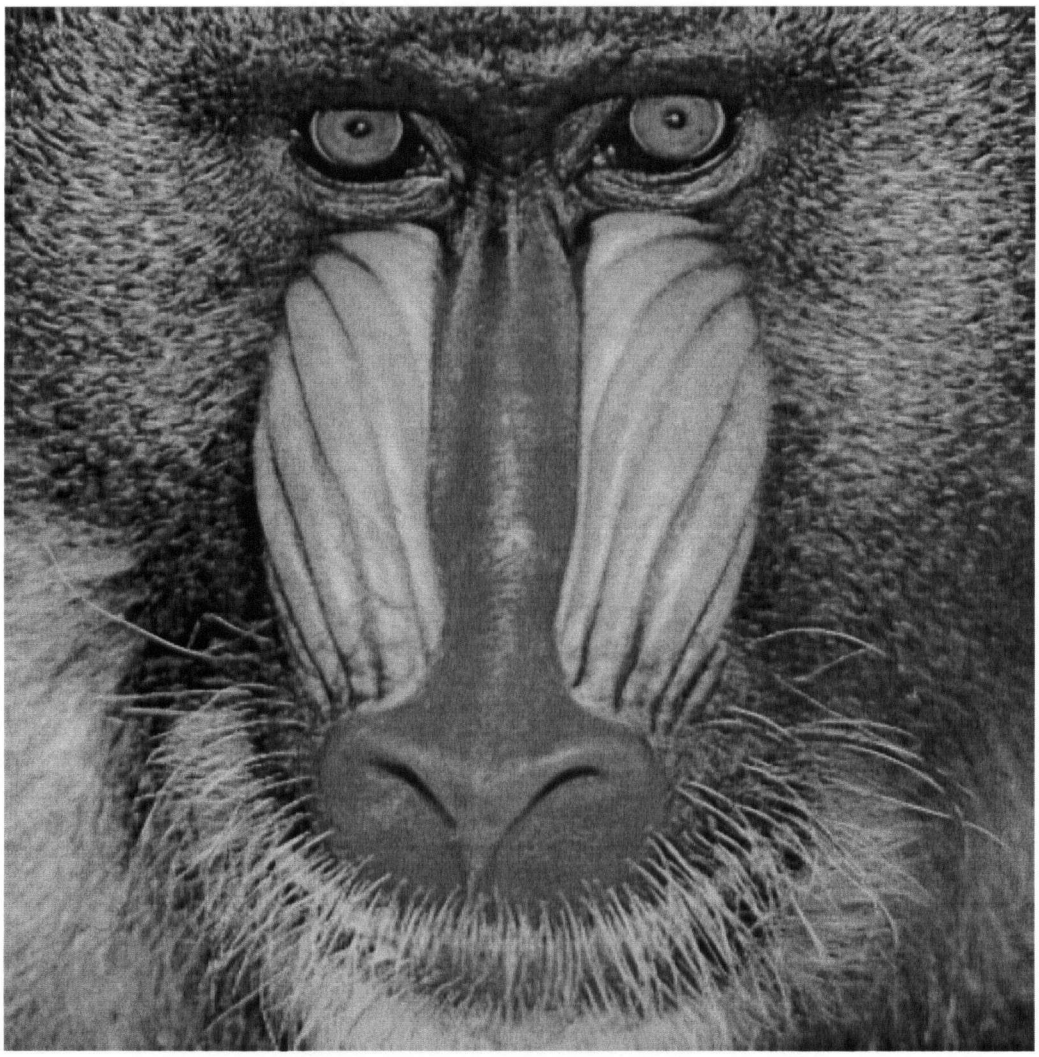


Figure 5.24 (c) Expanded by directional interpolation <u>with ridge interpolation</u>

Expansion ratio $R=1.5$. Notice the hair areas improved by the ridge interpolation.

5.5 Quantitative Evaluation of the Proposed Method

5.5.1 Curvature Evaluation

The isophote analysis in **5.1.3** shows that the proposed directional interpolation method yields very smooth isophotes in the interpolated images (see Figure 5.4 and Figure 5.5). This is also visually proven by the experimental results shown in **5.4**. In order to provide a quantitative evaluation for the isophote performance of the direction method, we calculate the average curvature of isophotes of the interpolated image as a measure of zigzagging in the interpolated images.

In our evaluation, we adopt the "minimum curvature" criterion, that is, the less the average curvature yielded by a certain interpolation method, the less zigzagged the image so interpolated. In reference [24], the average curvature is obtained by averaging the curvatures of all pixels in the interpolated image. In our study, we find that this approach has a drawback, which is explained below.

In an image that contains both smooth and edge areas, usually the average curvature of the smooth areas is greater than that of the edge areas. The isophotes along image edges usually have slow changes of orientations. That is to say, the curvatures of isophotes in the edge areas usually have small values, except for the few locations where the edges turn sharply. As a result, the average curvature of isophotes in the edge areas is usually low.

On the other hand, in the smooth areas, the isophotes very often appear as nested circles surrounding the local maxima and minima. At each local maxima or minima, the curvature has a very large (theoretically infinite) value. This property of isophote, combined with the small percentage the edge pixels occupy in the image, leads to the fact

that the average curvature of the edge areas is usually only a fraction of the overall average.

Because our purpose is to evaluate zigzagging artifacts, which mostly affect the visual appearance of the edge areas, a proper quantitative measure should be the average curvature of the edge pixels. Because curvature estimation usually has errors, curvatures of smooth pixels therefore act as strong interferences to what we are trying to detect. As a result, the evaluation based on the average curvature of all pixels is subject to very poor signal to noise ratio.

To demonstrate the high curvature values in the smooth areas, we show two curvature maps in Figure 5.25. This phenomenon is also proven by the data shown in the tables below. The curvature is estimated by the method proposed in **5.2.3**.

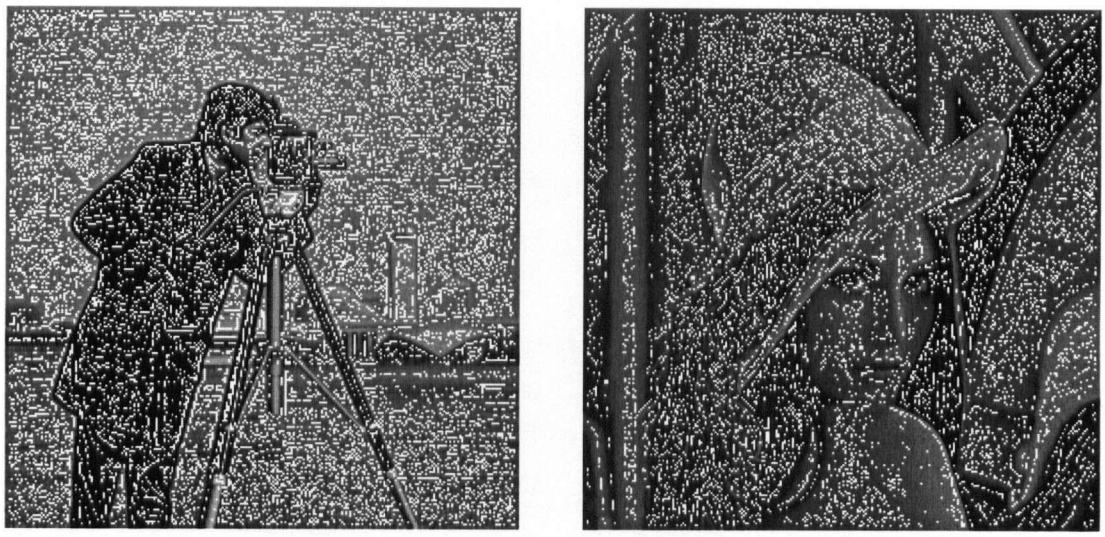
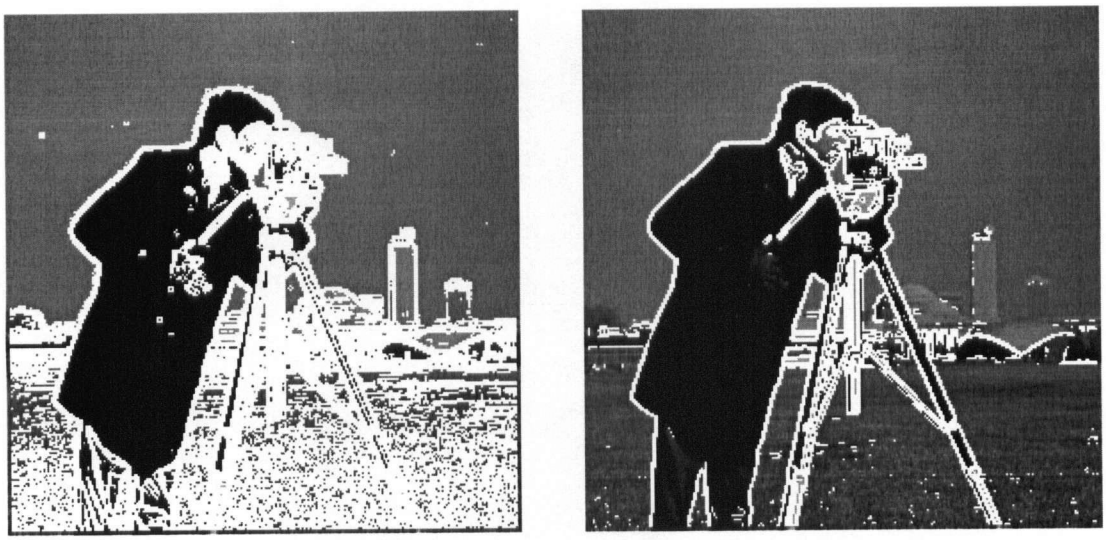


Figure 5.25 Curvature map of Camera Man and (reduced) Lena.

In the curvature maps, the highlighted pixels have estimated curvature values greater than 1.0 — note most of the edge pixels are not among them.

To avoid the interference caused by smooth pixels, in our evaluation, we calculate the average curvature for edge pixels only. The edge detection process is the same as in the proposed interpolation method.

Table 5-1, 5-2 and 5-3 show the average curvature of edge pixels detected under different threshold T_{grad}. Figure 5.26 is the edge map of (expanded) Camera Man corresponding to two different values of T_{grad}, showing the edge pixels involved in the curvature averaging. In Figure 5.26, the edge maps are reduced to fit the space.



(a) $T_{grad}=5$. (b) $T_{grad}=25$.

Figure 5.26 Edge maps with different gradient thresholds.

In Table 5-1, 5-2 and 5-3, the leftmost column is the gradient threshold T_{grad}. The rightmost column shows the percentage improvement (decrease) in average curvature by the directional method compared with the bi-linear/bi-cubic method.

Table 5-1. Average curvature of Camera Man for edge pixels.

	Bi-linear	Bi-cubic	Directional	Improvement
$T_{grad}=0$	0.679	0.668	0.671	1.1/-0.5
$T_{grad}=5$	0.338	0.355	0.314	7.1/11.6
$T_{grad}=15$	0.189	0.196	0.160	15.3/18.4
$T_{grad}=25$	0.151	0.152	0.119	21.2/21.7

Table 5-2. Average curvature of Lena for edge pixels.

	Bi-linear	Bi-cubic	Directional	Improvement
$T_{grad}=0$	0.652	0.648	0.649	0.5/-0.2
$T_{grad}=5$	0.237	0.252	0.208	12.2/17.5
$T_{grad}=10$	0.175	0.186	0.144	17.7/22.6
$T_{grad}=15$	0.143	0.151	0.109	23.8/27.8

Table 5-3. Average curvature of Tropic for edge pixels.

	Bi-linear	Bi-cubic	Directional	Improvement
$T_{grad}=0$	0.372	0.404	0.368	1.1/8.9
$T_{grad}=5$	0.148	0.161	0.120	18.9/25.5
$T_{grad}=10$	0.119	0.125	0.087	26.9/30.4
$T_{grad}=15$	0.102	0.105	0.069	32.4/34.3

In the above tables, $T_{grad}=0$ corresponds to the case that all pixels of the image are involved when calculating the average curvature. As we have predicted, the average curvature for $T_{grad}=0$ has the greatest value. Also, it is obvious from the results shown

that when all pixels are involved, the resulting average curvatures are often ineffective in reflecting the extent of zigzagging in the interpolated images. Compare the average curvature with $T_{grad}=0$ in all the three tables.

In our test, we gradually increase T_{grad}. With a higher T_{grad}, more interfering (smooth) pixels are filtered out, and edge pixels occupy a higher percentage in the averaged pixels. As a result, the average curvature becomes a better measure of the zigzagging artifact. In all the three tables, the higher the threshold, the more the directional method outperforms the traditional methods (see the percentage improvement). The data also proves our conclusion that the average curvature of the edge pixels is considerably lower than that of all pixels.

When edge pixels with a proper percentage are involved (refer to the last one or two rows of each table), the smoothness of the isophotes in the proposed directional interpolation method is consistently better than that of the traditional methods. This is an expected result, which is also matched by visual evaluation of the expanded images and their isophotes shown earlier in this chapter. Meanwhile, the bi-linear and bi-cubic methods are approximately comparable in most cases, with the bi-linear interpolation slightly better than the bi-cubic.

5.5.2 MSE Evaluation

As discussed in Chapter 3, the MSE-based measures are not consistently effective in evaluating the subjective quality of the interpolated images. Here, however, we still conduct a MSE evaluation of the proposed directional method. The reasons for carrying the MSE evaluation are as follows.

(1) MSE is still one of the most widely adopted methods of evaluation, and performed by many researchers in their study of image interpolation.

(2) By comparing the MSE with the corresponding visual quality of expanded images, we demonstrate the weakness of the MSE measures.

In the MSE tests, the original images are reduced and then expanded, so that they have the same size before and after the interpolations. The MSE is calculated by formula (2-15). In the following tables, we show MSE obtained from four different images, each expanded by 2.0 and 2.667 (8/3), through different interpolation methods.

Table 5-4. MSE of images expanded by 2.0

Method / Image	Bi-linear	Bi-cubic	Directional
Lena	4.267	4.006	4.099
Pepper	5.923	5.818	5.664
Zelda	3.790	3.746	3.792
Tropic	3.753	3.416	3.153

Table 5-5. MSE of images expanded by 2.667

Method / Image	Bi-linear	Bi-cubic	Directional
Lena	5.691	5.307	5.369
Pepper	7.255	6.993	6.857
Zelda	4.774	4.627	4.793
Tropic	5.293	4.737	4.224

As can be seen from the above tables, the proposed directional interpolation method does not always outperform the bi-linear and bi-cubic method in the MSE sense. This is

to some extent an expected result, in that the directional information is not designed to optimize the MSE measure. Similar results and conclusions are also drawn by other researchers (such as [24]) when comparing their proposed visually oriented methods to the traditional methods.

These results also show that the MSE measure is not very effective in evaluating the subjective quality of interpolated images. For an example, it can be seen from Table 5-4 and 5-5 that for image Lena, the MSE of bi-cubic interpolation is slightly better than that of the proposed method. But from subjective quality point of view, the proposed directional method yields by far superior results than the bi-cubic methods. The comparison of Lena expanded by 2.0 (corresponding to Table 5.4) can be seen in Figure 5.20. We show the images expanded by 2.667 (corresponding to Table 5.5) through different methods in Figure 5.27, which clearly demonstrates that, at least in this case, the MSE measure does not well match the extent of visual satisfaction.

5.5.3 Summary of the Directional Method

The experimental results and quantitative evaluations show that the proposed directional interpolation method is effective in removing zigzagging artifacts in the interpolated image. As a result, the subjective quality of the interpolated images are considerably improved, compared with the traditional methods.

The directional method yields smoother and more natural looking expanded images for different types of original images. For example, the Camera Man image has the special features of long and very sharp edges; Zelda and Pepper have some fine textures;

Lena has a mixture of edges, textures and details; Mandrill is a typical image that is rich of ridge objects.

Because a Mean Opinion Score (MOS) test is usually tedious, expensive and time-consuming, we do not conduct such a test. To evaluate subjective satisfaction of the proposed method, resulting images are shown to a number of viewers with various background and expertise, and feedbacks are collected. It is agreed by almost all viewers that images interpolated by our proposed method in virtually all cases give better subjective impressions, compared with images interpolated by traditional methods.

In our method, the directional interpolation is carried differently in the edge and ridge areas. In the edge areas, the interpolation parallelograms are along the isophotes; in the ridge areas, the parallelograms are perpendicular to the isophotes, i.e., along the ridgeline.

It is observed in our experiments that although the ridge pixels usually only occupy a low percentage, they are very important in the overall quality of the interpolated images. In many cases edges are accompanied by ridges of type (b) and (c) (refer to Figure 5.14). As a result, if the directional interpolation did not incorporate the ridges, the artifacts along many edges could not be completely suppressed. For results of the directional method with and without ridge interpolation, compare Figure 5.11 to Fig.5-19 (b), and Figure 5.12 and Figure 5.20 (b).

In the next chapter, we discuss another major degradation in image interpolation, the blurring effect, and introduce a contrast enhancement scheme to reduce the blurring.

Figure 5.27 (a) Expanded by 2.667 using bi-linear method.

Figure 5.27 (b) Expanded by 2.667 using bi-cubic method.

Figure 5.27 (c) Expanded by 2.667 using the directional method.

Chapter 6.
Contrast Enhancement in Image Interpolation

In this chapter, we introduce a contrast enhancement scheme to remove the blurring effects in the interpolated images. Section 1 discusses the blurring effects in image interpolation, and briefly reviews some recent work on contrast enhancement. Section 2 introduces a multi-band contrast enhancement scheme based on edge sharpening and unsharp masking (UM). Section 3 shows the experimental results and gives some discussions.

6.1 Overview

6.1.1 Adaptive Contrast Enhancement

As discussed in Chapter 3, traditional interpolation methods suffer from loss of contrast in the expanded images. This phenomenon is often referred to as the blurring effect. The blurring effect, apart from the zigzagging artifact, is another major degradation resulting from image expansion. Meanwhile, in most of the visually oriented interpolation methods reviewed in Chapter 3, the blurring problem is not specifically addressed and effectively solved. As a result, it is necessary in our research to study and design a method of contrast enhancement for the expanded images.

Contrast enhancement on its own is an in-depth topic, and numerous methods have been introduced to improve the image contrast in various applications. Generally, these

methods can be classified as (1) the traditional **global techniques** and (2) the recent **adaptive techniques**.

Among the traditional techniques are *histogram equalization (HE)*, *unsharp masking (UM)*, and the *linear contrast stretch* (also called windowing and leveling). Although the global methods are usually simple to implement and effective for some applications, they have some serious weaknesses, one of the most annoying ones being their sensitiveness to noise.

The recently proposed methods of contrast enhancement are generally divided into two categories – the spatial-domain methods and the frequency-domain methods. The spatial-domain methods, [55] - [59], employ content-adaptive UM or HE in order to provide adequate enhancement to image details while suppressing the noise level in smooth areas. The frequency-domain methods, [51] and [53], increase the image contrast by enhancing the high-frequency components. In particular, [53] introduces a multi-band contrast enhancement. In their method the image is divided into the low-, mid- and high-frequency bands. The mid- and high-bands are enhanced, whereas the low-band is not or little enhanced.

The adaptive contrast enhancement methods have improved the performance of the traditional methods. However, most of the methods mentioned above are designed for specific applications (e.g., medical images, or images after lossy de-compression), and are not particularly designed for enhancing the contrast of expanded images.

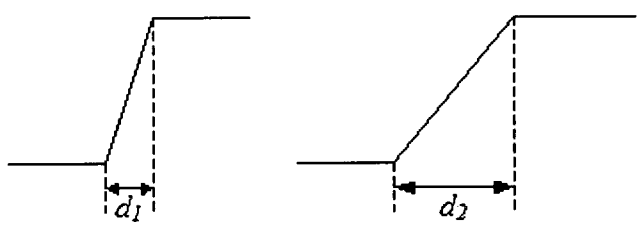
In order to design a contrast enhancement scheme to increase the visual sharpness of the expanded images, we adopt the idea of multi-band enhancement. In our method, an image is divided into two types of areas based on the nature of the content – the *edge*

areas (also called the high-contrast, high-activity, or detailed areas), and the *smooth areas* (also called the low-contrast, homogeneous, or monotone areas).

It is known that the edge areas are visually more important than the smooth areas in image perception. Also, in image expansion, the blurring effects are more harmful to the edge areas than to the smooth areas. As a result, our method mainly focuses on sharpening the edge areas.

6.1.2 Edge Sharpening in the Expanded Images

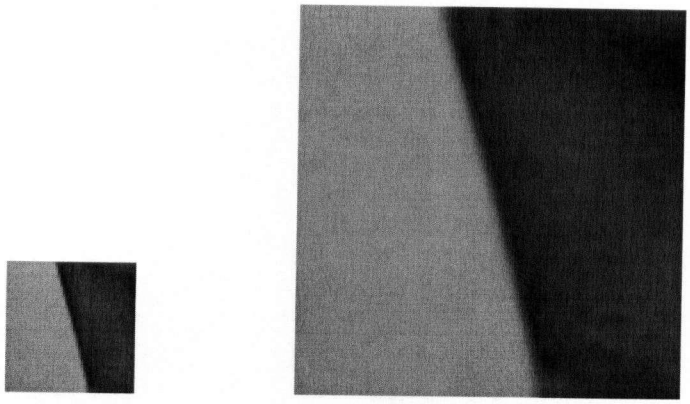
After expansion, the width of an edge is increased, while the difference in intensity across the edges remains almost unchanged. As a result, the sharp changes of intensity across the edges become gradual. This phenomenon is perceived as loss of contrast across the edge. See Figure 6.1 for illustration in a 1-dimensional case and 6.2 for a real example of a blurred edge.



(a) The original edge. (b) The expanded edge.

Figure 6.1 Expansion of a 1-d edge.

In the figure, d_1 and d_2 are the width of the edge before and after expansion, respectively. Approximately $\frac{d_2}{d_1}$ = expansion ratio.



(a) (b)

Figure 6.2 Blurring in the expanded edge.

In this figure, (a) is a part of Lena that has a typical image edge. (b) is the edge expanded by 3×3 using the directional method. Notice that the expanded edge does not suffer from zigzagging, but the visual appearance of blur still exists.

The proposed directional interpolation, described in Chapter 5, removes the zigzagging artifacts by providing smooth isophotes in the interpolated image. However, it does not enhance the contrast of the interpolated edges. Therefore, after the directional interpolation proposed in Chapter 5, a separate process of contrast enhancement is needed so that contrast comparable to the original image is restored.

Among the recent interpolation methods reviewed in Chapter 3, references [12] and [25] proposed an **interpolate-then-sharpen** approach. In these two-phased schemes, the original image is first interpolated using the visually oriented method; the interpolated image then undergoes an edge-sharpening process that enhances the contrast in the edge areas. To carry the edge sharpening, the pixel intensity values of the interpolated edges are changed, so that the profile of the edges is steeper than those initially interpolated. This process is explained in the one-dimensional case as follows (refer to Figure 6.3).

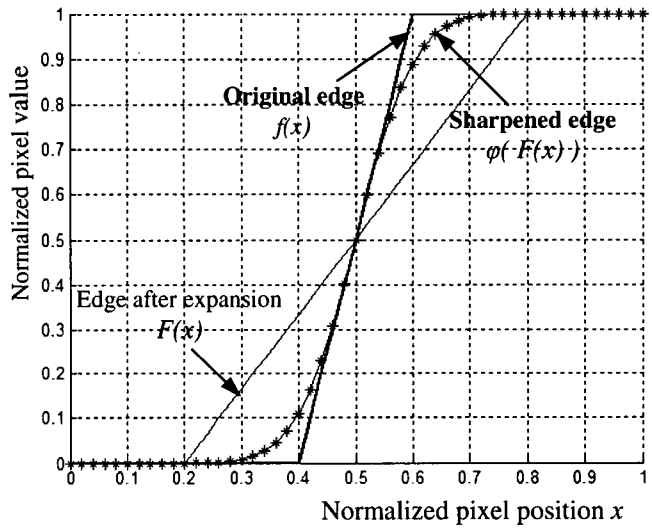


Figure 6.3 Sharpening a 1-d expanded edge.

In the figure, all functions are normalized in both x and y directions.

In Figure 6.3, the expanded edge $F(x)$ is less sharp than the original edge $f(x)$. A roll-down function $\varphi(t)$ is applied on $F(x)$, resulting in the modified edge $\varphi(F(x))$. If function $\varphi(t)$ is properly designed, the central part of the modified edge $\varphi(F(x))$ (around $x=0.5$) becomes steeper then the parts on both sides, and thus a sharper edge between the smooth areas (areas around $x=0$ and $x=1$) results.

In [12], the edge-sharpening function (called the "weighting function") is

$$\sigma(x) = \frac{1}{1+e^{-rx}} \qquad (6\text{-}1)$$

In (6-1) r is a free parameter that controls the shape of $\sigma(x)$, i.e., the extent of sharpening. See Figure 6.4 for curves of $\sigma(x)$.

In [25], we proposed an edge-sharpening function (called the "edge re-shaping function" in that reference):

$$\varphi(x) = \begin{cases} 0 & x < 0 \\ \dfrac{x^2}{2x^2 - 2x + 1} & 0 \le x \le 1 \\ 1 & x > 1 \end{cases} \qquad (6\text{-}2)$$

The curve of (6-2) can be seen in Figure 6.5 (the curve with $a=2.0$).

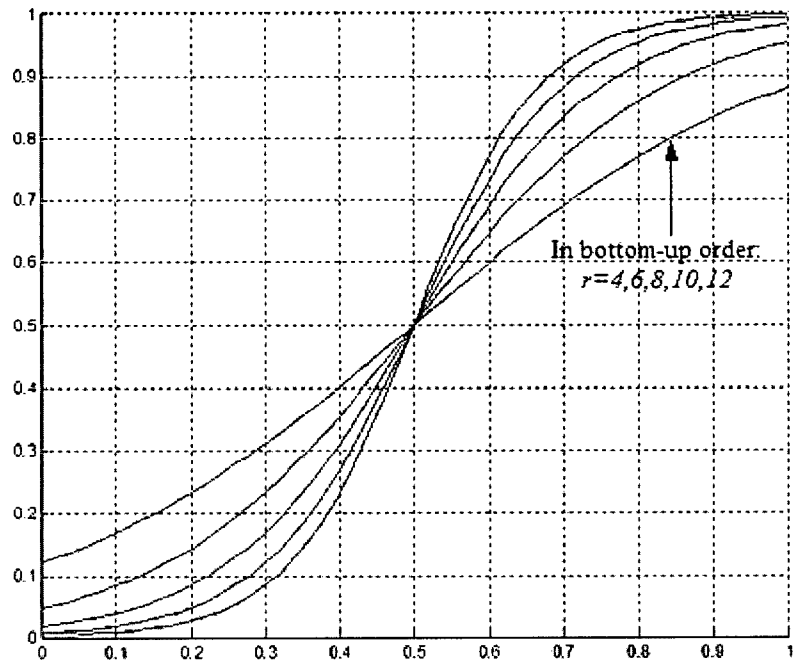


Figure 6.4 $\sigma(x)$ with different values of r.

References [12] and [25] show that the edge-sharpening schemes based on the above functions improve the contrast of the expanded images, especially in the edge areas. As a result, the blurring effects in the expanded images are reduced.

However, the edge-sharpening functions of (6-1) and (6-2) have their own disadvantages. In the following section, we study the properties of a good edge-sharpening function, and introduce a family of edge-sharpening functions. Based on the introduced functions, we design the corresponding edge sharpening scheme.

6.2 The Proposed Image-Sharpening Scheme

6.2.1 The Edge-Sharpening Function

Edge-sharpening functions employed by [12] and [25] ((6-1) and (6-2), respectively) are both roll-down functions, which satisfy the general requirement of sharpening edges. However, in order to achieve better performance, further requirements on the edge-sharpening function should be added. In our study, we found the following properties are desirable for an edge-sharpening function.

Assume $\varphi(x)$ is a normalized edge-sharpening function defined on $0 \leq x \leq 1$.

(1) $\varphi(x)$ is continuous.

(2) $\varphi(0) = 0, \varphi(1) = 1$.

(3) $\varphi'(0) = \varphi'(1) = 0$.

(4) $\varphi(x)$ does not have overshooting around $x=0$ and $x=1$.

(5) $\varphi(x)$ has a parameter that controls the steepness in the central part of $\varphi(x)$. In particular, the parameter could be adjusted so that

$$\varphi'(0.5)=R \quad (R>1 \text{ is the expansion ratio}) \qquad (6\text{-}3)$$

Restriction (1) is obvious. (2) and (3) guarantee that the sharpened edges smoothly connect to the smooth areas on both its sides. (4) is important for the sharpened edges to be ringing-free, which is a concern especially in the frequency-domain methods. Property (5) is needed to provide different extent of the edge-sharpening effect under different circumstances. The particular requirement of equation (6-3) is explained as follows (refer to Figure 6.3 for illustration).

Assuming the profile of the original edge is linear with unit steepness, i.e., $f(x)=x$, the expanded edge is (approximately) $F(x)=x/R$ (refer to Figure 6.3). Applying the edge-sharpening function yields the sharpened edge

$$F_d(x) = \varphi(F(x)) = \varphi(\frac{x}{R}) \qquad (6\text{-}4)$$

If, after edge sharpening, we require that the central part (around $x=0.5$) of $F_d(x)$ have approximately the same steepness as that of the original edge, i.e.,

$$F_d'(x) = 1, \quad for\ x = 0.5, \qquad (6\text{-}5)$$

we could then show that

$$\varphi'(x) = R \cdot F_d'(x) = R, \text{ for } x = 0.5$$

Function $\sigma(x)$ in (6-1) satisfies property (1), (4) and (5), but not (2) and (3).

The edge-sharpening function (6-2) we proposed in [25] satisfies restrictions (1) through (4), but not (5), meaning that it does not have a free parameter by which the extent of edge sharpening can be adjusted.

We can show that function (6-2) is a member of the function family

$$\varphi_a(x) = \begin{cases} 0 & x < 0 \\ \dfrac{x^a}{x^a + (1-x)^a}, & 0 \le x \le 1 \\ 1 & x > 1 \end{cases} \qquad (6\text{-}6)$$

In (6-6) the parameter a is a positive real number. Function (6-2) is a special case of (6-6) with $a=2.0$. Function $\varphi_a(x)$ has found applications in other fields of studies, such as in the discrete Markov analysis. In this research we introduce it as the edge-sharpening functions family.

We can prove that $\varphi_a(x)$ satisfies all the restrictions (1) through (5). Besides, we can also prove that $\varphi_a(x)$ has other properties that are favorable for edge sharpening, some of them are listed below.

(6) $\varphi_a'(0.5) = a$.

(7) $\varphi_a[\varphi_b(x)] = \varphi_b[\varphi_a(x)] = \varphi_{ab}(x)$, i.e., the superposition property.

(8) When $a=1$, $\varphi_a(x)=x$, and when $a \rightarrow \infty$, $\varphi_a(x) \rightarrow$ the ideal step function.

Property (6) reveals that parameter $\boldsymbol{a}$ is a direct indication of the steepness of the sharpened edge. Combining property (6) and equation (6-3), it is concluded that when sharpening expanded edges using $\varphi_a(x)$, it is appropriate to choose

$$a = \textit{the Expansion Ratio} \qquad (6\text{-}7)$$

Based on property (7), if $\varphi_a(x)$ is applied repeatedly, the overall effect equals that of applying $\varphi_a(x)$ once with the value of $\boldsymbol{a}$ being the product of all the other a's.

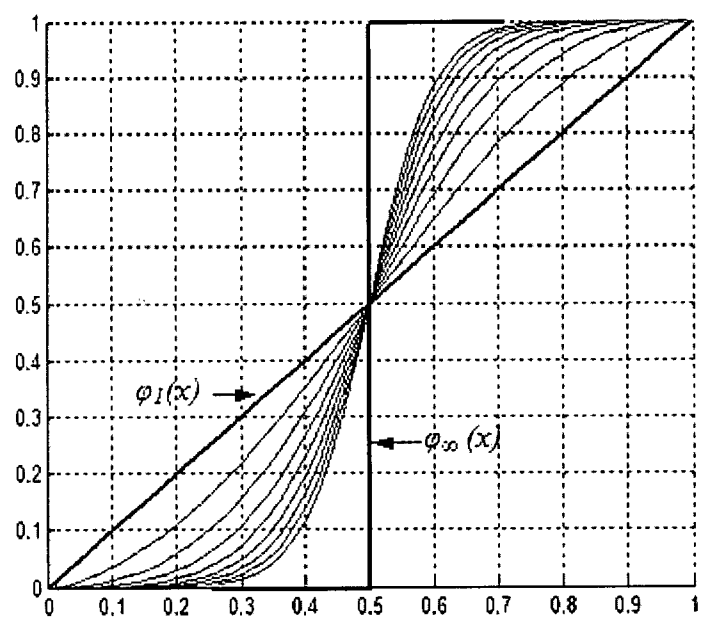


Figure 6.5 Curves of $\varphi_a(x)$ with different values of a

Starting from $\varphi_1(x)$: $a=1, 1.5, 2, 2.5, 3, 3.5, 4, 4.5, 5, \infty$

Property (8) implies that when $\boldsymbol{a} = 1$, $\varphi_a(x)$ does not change the original edge $f(x)$ (no sharpening); with a = ∞, $\varphi_a(x)$ changes the original edge into an ideal step edge, regardless of its original shape. The curves of $\varphi_a(x)$ are shown in Figure 6.5.

Based on the edge-sharpening function derived, we now design the edge-sharpening algorithm below.

6.2.2 The Edge-Sharpening Algorithm

In our edge-sharpening scheme, the edges in the interpolated image are first detected; then we apply the proposed edge-sharpening function to the detected edges. This process is illustrated in Figure 6.6.

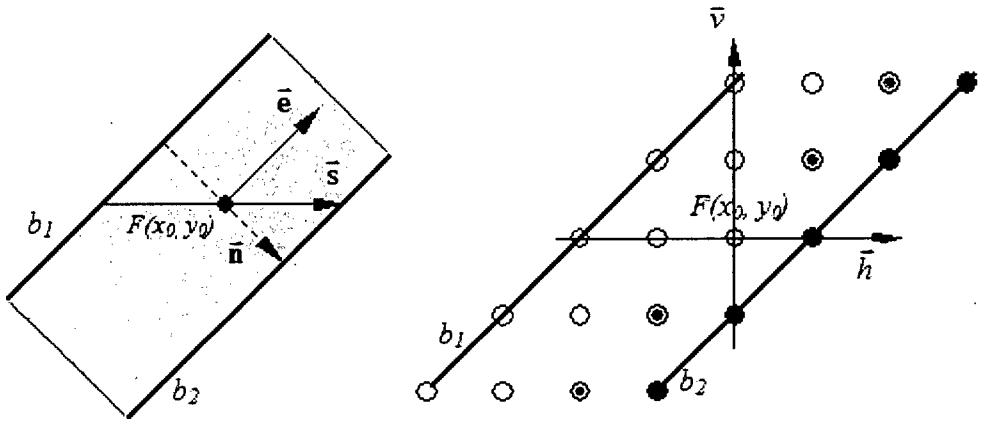
Figure 6.6 (a) shows a small region of an edge area, and we assume that within such a small region the isophotes are straight lines parallel to the edge direction. Based on this assumption, the following properties can be derived or assumed:

(1) The direction of the edge is the same as that of the isophotes. The boundary lines of this region, ***b1*** and ***b***2 in Figure 6.6 (a), are isophotes themselves.

(2) The edge direction is $\vec{\mathbf{e}}$, and the normal vector of the isophotes is $\vec{\mathbf{n}}$.

(3) The pixel value on boundary line (isophote) ***b1*** is V_{min}, and on ***b2*** is V_{max}. Pixel values increase monotonically from V_{min} to V_{max} along the normal direction $\vec{\mathbf{n}}$.

(4) $F(x_0, y_0)$ is a pixel whose value is to be changed during the edge sharpening.



(a) (b)

Figure 6.6 Applying the edge-sharpening function on edge pixels.

In the ideal case, the edge-sharpening function $\varphi_a(x)$ should be applied along direction $\vec{\mathbf{n}}$ across the edge area, so that the profile of the edge is sharpened (also refer to Figure 6.3). The edge-sharpening process so based would be carried as follows:

(1) For all pixels on the normal line that passes through point $F(x_0, y_0)$, their values are first normalized to 0 to 1 to lie in the range of 0 to 1.

(2) The normalized pixel values are sharpened using

$$\hat{U}_i = \varphi_a(U_i) \qquad (6\text{-}8)$$

(3) The sharpened pixels values are then de-normalized so that the values of the original V_{min} and V_{max} are re-obtained.

However, in the actual interpolated lattice, because the edge direction $\vec{\mathbf{e}}$ is arbitrary, the grid pixels are usually not aligned in the normal direction $\vec{\mathbf{n}}$, and thus the above process cannot be directly applied.

Based on assumptions (1) and (3) above, we can prove that $\varphi_a(x)$ can be applied on a line passing through $F(x_0, y_0)$ from any direction and the same result will be achieved. In our algorithm, we apply $\varphi_a(x)$ in two orthogonal directions, horizontal and vertical but with $a = \sqrt{the\ Expansion\ Ratio}$ for each. By virtue of the superposition property of $\varphi_a(x)$, the overall effect of sharpening is equivalent to *a=the Expansion Ratio*. See Figure 6.6b for illustration of the above separable sharpening process.

6.2.3 Edge Detection

Both our edge-sharpening algorithm and our interpolation method, as described in Chapter 5, require edge information of the expanded image, thus the edge information needed for edge sharpening is already available. However, in our edge-sharpening scheme, we detect the edges again although it seems unnecessary. This is because in our experiments, we found that for the sharpening purpose, smoothing the interpolated image before detecting its edges yields better results.

After proper smoothing, edge detection yields edge areas with better connectivity and that are less sensitive to noise. This is highly desirable since our edge-sharpening algorithm requires the edge pixels to be consecutively aligned (see Figure 6.6 (b)). After smoothing, the edge map changes, and therefore has to be detected again.

As our sharpening algorithm has the advantage that it only requires the location information of edges and not their orientation, edge detectors that are easy to implement (and usually less accurate in estimating the edge's direction) can be used, e.g., the 4-neighbor edge detector or the classic Sobel operator. In our algorithm, we still use the improved Sobel operator, the same as used in the interpolation algorithm.

In the preliminary smoothing, we use the Gaussian filter with $\sigma=1.0$.

6.2.4 The Contrast Enhancement Scheme

As described above, for the edge areas in the interpolated image, the edge sharpening operation is performed to enhance the contrast. For the smooth areas of the image, we use the traditional unsharp-masking (UM) to increase the contrast. The UM kernel employed in our algorithm is

$$\begin{bmatrix} a & a & a \\ a & b & a \\ a & a & a \end{bmatrix}, \quad a = \frac{c-1}{18c-9}, b = \frac{10c-1}{18c-9}, \quad 0.5 < c \le 1. \qquad (6\text{-}9)$$

In (6-10), the lower the value of c, the stronger the sharpening effect. If $c=1$, the image is unchanged after the UM operation. Because strong UM operation tends to raise the noise level in the output image, we usually avoid strong UM by choosing $c \geq 0.8$.

The flow diagram of our contrast enhancement scheme is shown in Figure 6.7.

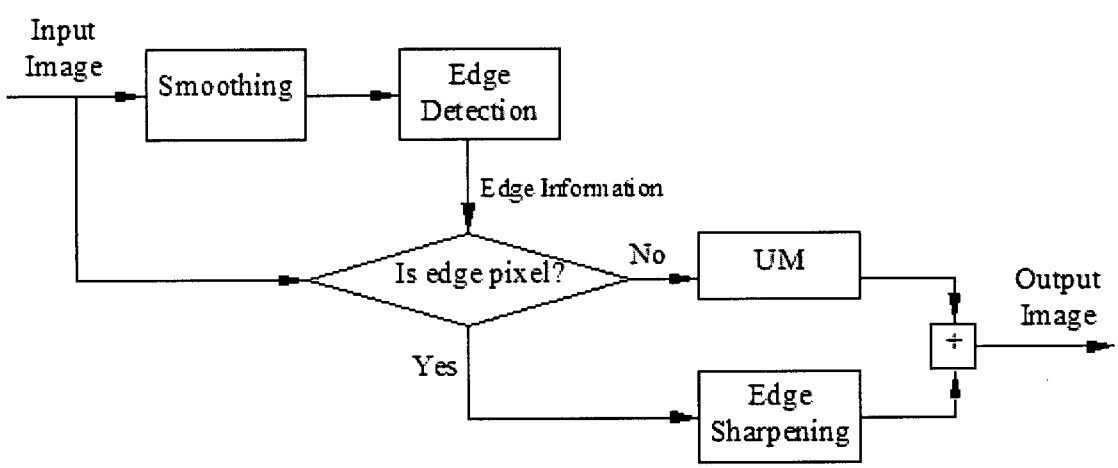


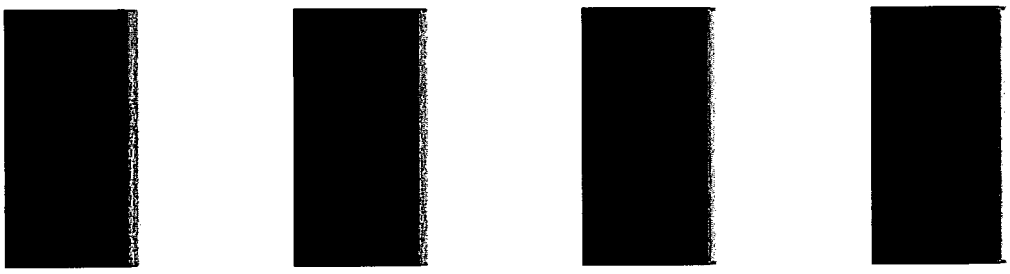
Figure 6.7 Flow diagram of the proposed contrast enhancement scheme

6.3 Experimental Results and Conclusions

We first test the performance of the proposed edge-sharpening algorithm on simulated (artificially created) edges. Shown in Figure 6.8 are some of the results obtained.

In Figure 6.8 (a), (b), and (c), from left to right are the original blurred edge, the edge sharpened with $a=2$, $a=4$, and $a=8$. The original blurred edge is linearly interpolated from an ideal step edge by a factor of 8, and then rotated by a certain angle. In Figure 6.8, edges of 90°, 65°, and 45° are shown.

The results show that our edge-sharpening scheme is effective in thinning the edges. For different angles of edge, the sharpening yields satisfying results without introducing noticeable artifacts (such as ringing or zigzagging) to the sharpened edges. With *a=the Expansion ratio* ($a=8.0$, the rightmost image of all 3 angles), satisfying visual sharpness is achieved in the sharpened edges.



(a) 90° edge. The edge is on the right side of each black area

(b) 65° edge

(c) 45° edge

Figure 6.8 Sharpening the simulated edges.

We then apply the proposed contrast enhancement algorithm to images interpolated by the directional method proposed in Chapter 5. In Figure 6.9 through 6.11, we show the results of image Pepper, Zelda, and Lena. The images are initially expanded by a factor of 2.0×2.0 using the directional method; then their edges are sharpened, and the non-edge areas are enhanced by unsharp masking. The original images are also shown for comparison purposes.

The original images are first down-sampled to 0.5×0.5 of their original sizes; they are then expanded to their original sizes. For edge sharpening, the sharpening parameter a has the value of $\sqrt{2}$ horizontally and vertically, resulting in an overall strength of $a=2.0$. For the UM, we choose a low strength with $c=0.8\sim0.9$.

From our experimental results, it is seen that the proposed scheme of contrast enhancement effectively raises the visual contrast level of the expanded images. The blurring effects resulting from expansion are largely alleviated by the sharpening scheme.

The proposed image sharpening method is not restricted to sharpening images expanded by our proposed directional method. It is also applicable to other images that have expansion-related blurring. However, it is an expected conclusion, and also proven

by our experiments, that the proposed edge-sharpening method does not equally improve the subjective quality of images expanded by the traditionally interpolated methods. The reason is that the proposed edge-sharpening method does not remove the zigzagging artifacts associated with the traditional interpolation methods. In fact, it is observed in our experiments that after edge sharpening the zigzagging artifacts could be worsened in some cases. In order to show the effect of the proposed sharpening method on images expanded by traditional methods, we apply the sharpening method on Lena after it is expanded by bi-cubic interpolation. The result is shown in Figure 6.12. As can be seen, although the edges indeed become sharper, the overall visual quality of the image is hardly improved by sharpening.

It is also expected and observed that with inappropriately strong strength of edge sharpening, e.g., $a=6$ for images expanded by 2, over-sharpening could result. One of the worse cases of over-sharpening is the cartooning effect, which is also one of the drawbacks in some of the edge-directed interpolation methods. For an example of over-sharpened images, see Figure 6.13.

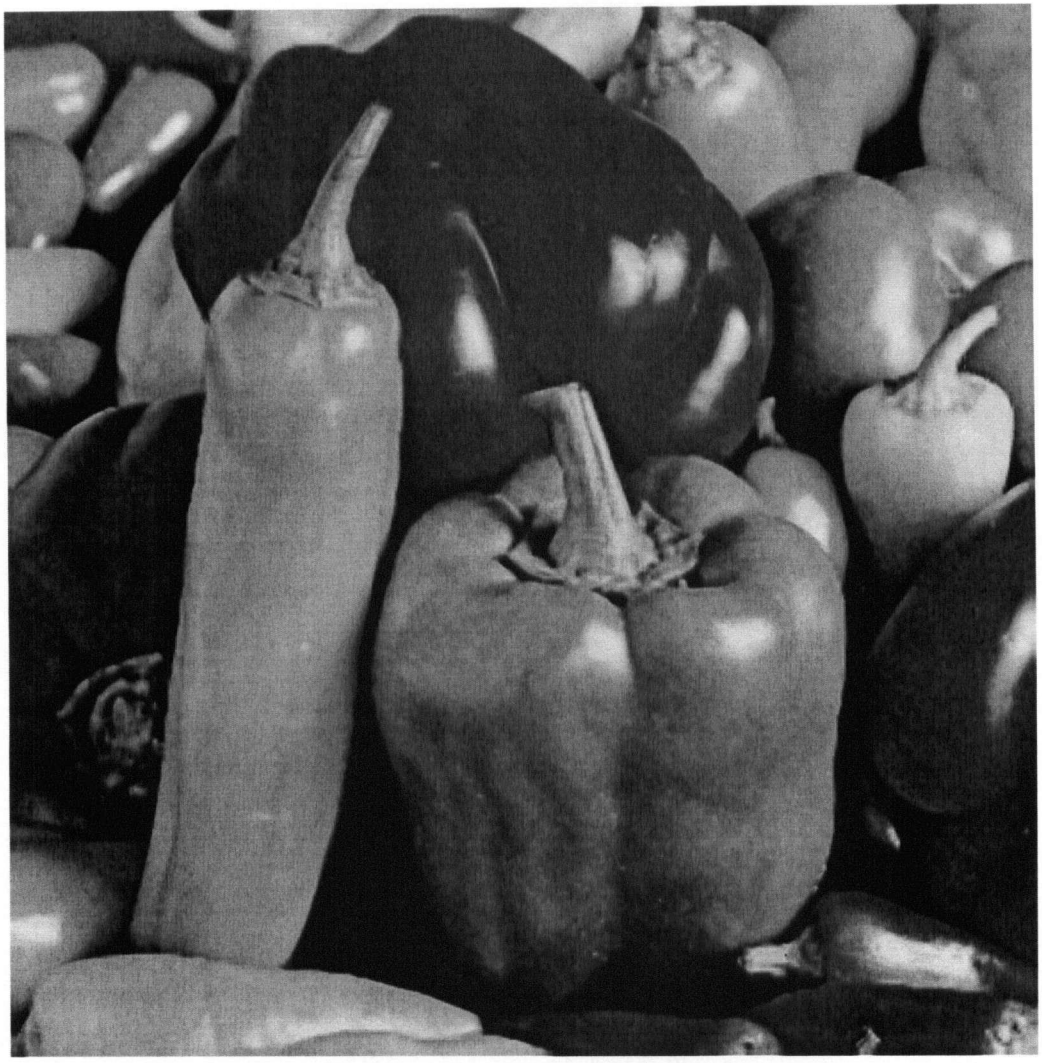


Figure 6.9 (a) Pepper expanded without contrast enhancement

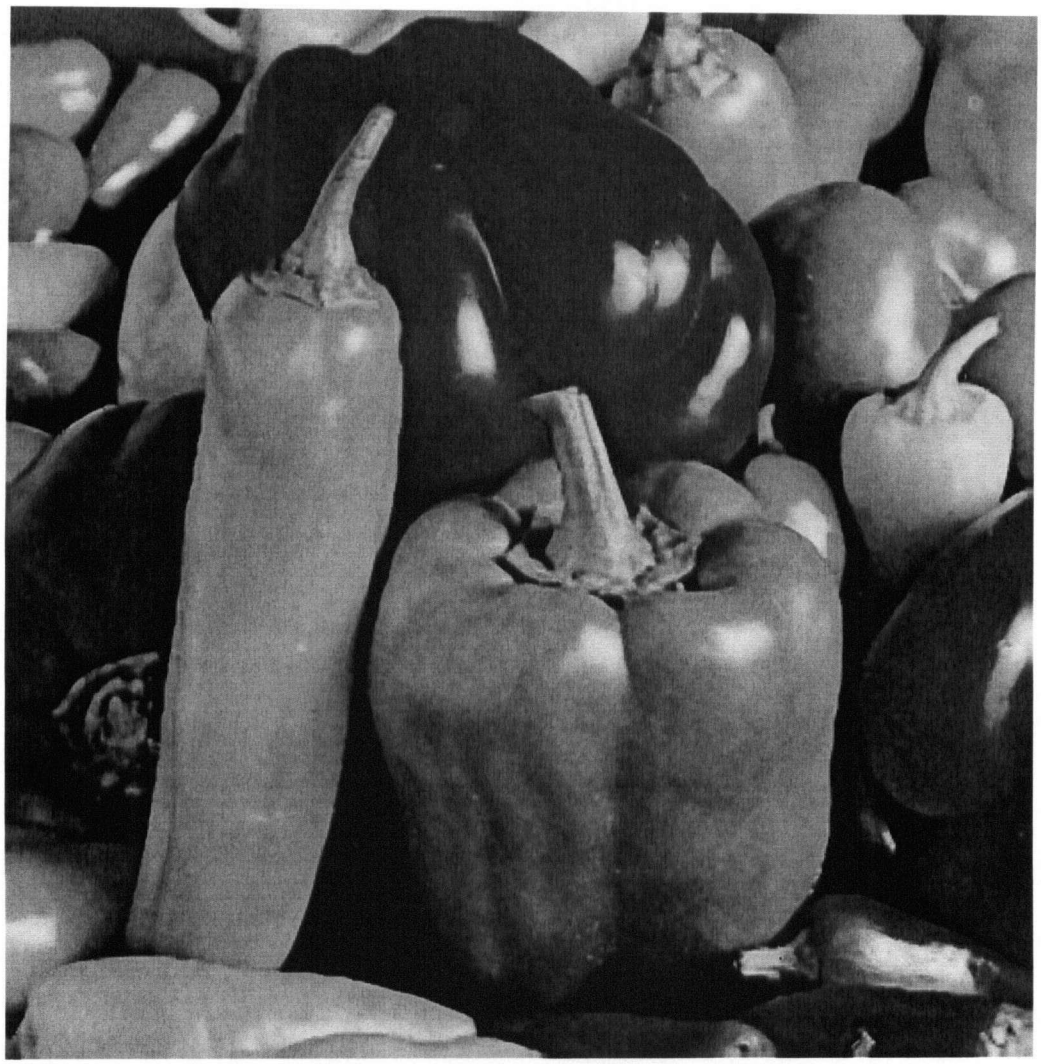


Figure 6.9 (b) Pepper sharpened after expansion

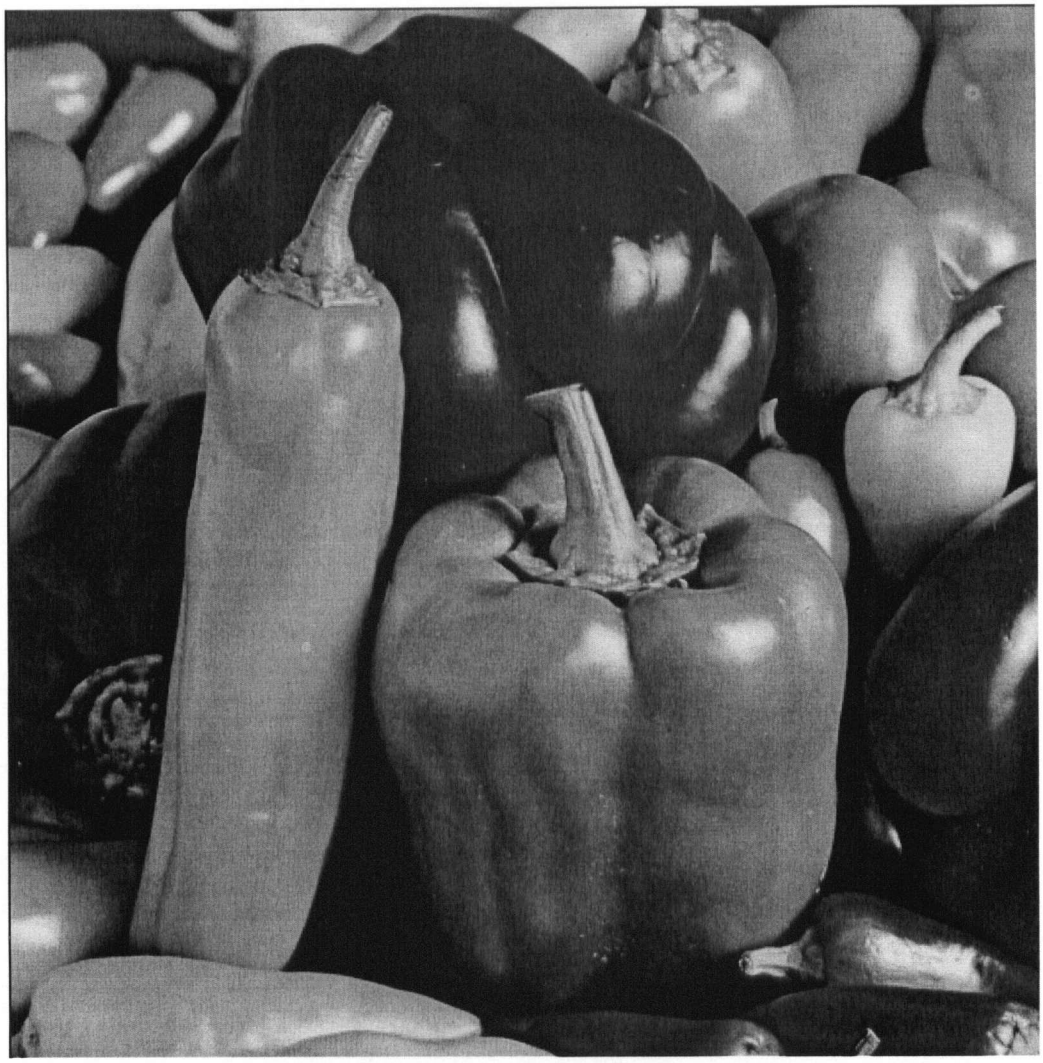


Figure 6.9 (c) Original image Pepper

Figure 6.10 (a) Zelda expanded without contrast enhancement

Figure 6.10 (b) Zelda sharpened after expansion

Figure 6.10 (c) Original image Zelda

Figure 6.11 (a) Lena expanded without contrast enhancement

Figure 6.11 (b) Lena sharpened after expansion

Figure 6.11 (c) Original image Lena

Figure 6.12 Lena sharpened after expansion by bi-cubic method

Edge sharpening is performed with a=2.0. No UM is carried. Notice that the zigzagging effects are not alleviated, in some cases even exaggerated. The overall subjective quality is hardly improved.

Figure 6.13 An example of over-sharpening

Zelda expanded by 2 and sharpened with $a=10$. Notice the cartooning effect resulting form over-sharpening.

Chapter 7.
Conclusions and Discussions

In section 1, we summarize the proposed directional interpolation method and the contrast enhancement scheme. Section 2 summarizes the contribution of this thesis. In section 3, we give some suggestions of future work.

7.1 Summary of the Proposed Interpolation Method

In our research of the visually oriented image interpolation, the two major degradations in the traditional methods, the zigzagging artifact and the blurring effects, are solved in two separate steps, the directional interpolation and image sharpening.

In Chapter 5, we introduced the directional interpolation method. In the proposed method, an image is divided into two types of areas, the smooth areas and areas that contain orientation information. For the orientation-featured areas, we further divide them into edge and ridge areas, in both of which the interpolation grids are direction-adaptive parallelograms. Meanwhile, based on different isophote models of the edge and ridge areas, different mechanisms of estimating the orientation are designed and employed for those areas.

The proposed directional method is based on visual satisfaction, and belongs to the visually oriented family that includes other such methods reviewed in Chapter 3. In the

following, we briefly compare the proposed method and other visually oriented methods and discuss the difference between them

Compared with the edge-directed methods, the proposed method does not require precise detection of the very thin image edges. As our isophote analysis reveals, usually within a fairly wide area along the edges, the isophotes are nearly parallel and straight. The directional interpolation method smoothes all these isophotes, including the very thin edge line detected in the edge-directed methods. By doing so, the zigzagging artifacts are removed more thoroughly. Also, the edge-directed methods are subject to the problem of detecting ridges, because of the limitation of the gradient-based edge detection.

Compared with the orientation-adaptive methods, the proposed interpolation method is more flexible in that it is applicable to arbitrary interpolation ratio, including $R<1$ which corresponds to image reduction. One important reason for this flexibility, seemingly only technical, is that the proposed directional interpolation is in nature a generalized bi-linear interpolation, which is largely a flexible method. The method proposed in [15] requires that spatial pattern of the expanded grid be identical to that of the original grid in order to apply the orientation information, thus causing the 2^n restriction.

It is also desirable to show the experimental results from other visually oriented methods for comparison purpose. However, due to difficulties in obtaining these results with proper quality, they are not included in this thesis.

The proposed directional method employs a one-pass algorithm, which is more computationally economic, compared with the iterative approaches adopted by the level set-based methods.

In Chapter 6, we introduce a method for sharpening the expanded images. In this method, the edge areas are sharpened by a proposed edge-sharpening function, and the smooth areas are unsharp masked with light strength. As reviewed in Chapter 3 and 6, the blurring effect in image expansion is not properly addressed in most of the recent methods, and is not specifically solved.

In studying the edge-sharpening functions, we raised a set of properties desirable for edge sharpening. Based on the above analysis, we suggested a family of roll-down functions for edge sharpening, and developed the corresponding algorithm. Our experiments show that the so-based image-sharpening scheme is effective in enhancing the contrast of the expanded images.

The computational cost of the proposed directional method is low to moderate. Our estimation concludes that the directional interpolation (excluding the image sharpening) takes about 6 times the computation required for the traditional bi-linear method, and less than that of the 4-point bi-cubic interpolation for the case of arbitrary interpolation ratio. Without optimization, the computational time of image Camera Man (256×256) is about 0.5 second. Such modest computational complexity enables the proposed algorithm to have the potential of being implemented in real time, with the help of a modern graphics-associated system.

Although throughout this thesis we focus our effort on interpolation of gray scale images, the proposed interpolation and contrast enhancement scheme can both be extended to color images. In some of our experiments, we interpolated the red, green and blue components of color images using the proposed directional methods, resulting in the interpolated color images. We also tested the proposed image-sharpening scheme on

color images using the same approach. The results show that images interpolated by the proposed method are very visually impressive. Although we only tested a simple method of extending the proposed method to color images, it is proved that the visual advantages of proposed directional method are also applicable to color images, similarly to the case of gray scale images.

7.2 Contribution of This Thesis

The main contributions of our research are summarized as follows.

- After surveying the visually oriented interpolation methods, we classify the recently introduced methods into three categories: **edge-directed**, **orientation-adaptive**, and **level set-based**. For each category, the representative methods are reviewed and discussed.
- We propose a method of analytically calculating isophotes for bi-linear and bi-cubic interpolation methods, and explain the zigzagging artifacts using the reconstructed isophotes. In particular, we show that in the traditional bi-linear methods, the reconstructed isophotes are piecewise hyperbolae, and that by employing interpolation grid that adapts to the orientation of isophotes, the resulting zigzagging can be reduced.
- Based on the isophote analysis, we proposed a novel interpolation method. Our proposed method is a generalized bi-linear interpolation, in which the interpolation grid adapts to the local orientation of isophotes. By isophote derivations, curvature evaluation and experimental results, we showed that the proposed interpolation

method yields smoother isophotes in the interpolated images, thus eliminating the zigzagging artifacts.

- We studied a type of visually important image features that have strong orientation information, ridgelines. We show that the gradient-based estimators, adopted by many recently introduced interpolation methods, are inefficient in detecting ridges. We proposed an ellipse-based isophote model for ridgelines, and accordingly developed an algorithm that yields smooth and artifact-free ridgelines.
- In order to remove blurring effects after image expansion, we proposed a content-dependent enhancement scheme for expanded images. We introduced a family of roll-down functions that we show to have desirable properties for edge sharpening, and developed an adaptive contrast enhancement algorithm based on the proposed edge-sharpening functions.

7.3 Suggestions for Future Work

From our study and experiments, it is noticed that there are still some aspects in which the proposed directional interpolation method can be further improved. Because of the limit of time and scope, these improvements cannot all be completed within this research. However, below we summarize some of the questions and problems that still exist in the proposed method, and give suggestions for future work.

In the proposed directional interpolation method, the accuracy and reliability of estimating the orientation of isophotes can still be improved. In our algorithm, we employ the improved Sobel operator to predict the orientation, and implement the curvature

thresholding to prevent false estimations. This approach proves to be effective and yields satisfying results. Meanwhile, in some cases the estimation errors are still not completely eliminated, and some of them lead to erroneously interpolated pixels. Although these cases are very rare and yield little noticeable degradations, it is still desirable that these errors are corrected. In the future work, it is suggested that more study on estimating the orientation of image objects be carried, so that the performance of the directional interpolation is further improved.

The directional interpolation and image sharpening procedure are currently separate from one another. We realize that in a typical application of image expansion, in which both expansion and sharpening are to be carried, it is feasible that these two steps are combined. A unified *sharpen-when-interpolate* process is more concise and computationally economical.

The proposed idea of adapting the interpolation grid to the local orientation is in nature not restricted to the parallelogram grid proposed in Chapter 5, which originated from the bi-linear grid. For example, the orientation-adaptive grid can also be applied to the bi-cubic grid, resulting in the directional bi-cubic interpolation. A probable advantage of the orientation-adaptive cubic interpolation is the improved MSE compared with the linear-based methods.

In our experiments, we tested a method that performs the directional interpolation on the R, G and B components separately, resulting in interpolated color images. This is rather a proof of applicability than a practical scheme. For practical applications, a more efficient method that utilizes the correlation of different color components needs to be developed. One such method is to use the YUV representation of color images, instead of

the RGB. Because usually it is the intensity component (Y) that contains most visually important information in a color image, it is reasonable to apply the proposed interpolation method only on the Y component, while interpolating the less important U and V components using other methods that are computationally simple.

Of practical importance, the structure of the algorithm, and its implementation as well, are still to be optimized. The optimization of the algorithm is especially important for possible future applications in video processing.

Appendix

A-1. Coefficient Matrix of Cubic Interpolation

The following is the matrix E in equation (4-13). See 4.3.2. In matrix $\boldsymbol{E}$, the parameter a is that of Keys' cubic spline.

Column 1 through 8

Row								
1	a^2	$-5a^2$	$8a^2$	$-4a^2$	$-5a^2$	$25a^2$	$-40a^2$	$20a^2$
2	a^2+2a	$-5a^2-10a$	$8a^2+16a$	$-4a^2-8a$	$-4a^2-9a$	$20a^2+45a$	$-32a^2-72a$	$16a^2+36a$
3	$-a^2-2a$	$5a^2+10a$	$-8a^2-16a$	$4a^2+8a$	$5a^2+9a$	$-25a^2-45a$	$40a^2+72a$	$-20a^2-36a$
4	$-a^2$	$5a^2$	$-8a^2$	$4a^2$	$4a^2$	$-20a^2$	$32a^2$	$-16a^2$
5	a^2+2a	$-4a^2-9a$	$5a^2+12a$	$-2a^2-4a$	$-5a^2-10a$	$20a^2+45a$	$-25a^2-60a$	$10a^2+20a$
6	a^2+4a+4	$-4a^2-17a-18$	$5a^2+22a+24$	$-2a^2-8a-8$	$-4a^2-17a-18$	$16a^2+72a+81$	$-20a^2-93a-108$	$8a^2+34a+36$
7	a^2-4a-4	$4a^2+17a+18$	$-5a^2-22a-24$	$2a^2+8a+8$	$5a^2+19a+18$	$-20a^2-81a-8$	$25a^2+105a+108$	$-10a^2-38a-36$
8	$-a^2-2a$	$4a^2+9a$	$-5a^2-12a$	$2a^2+4a$	$4a^2+8a$	$-16a^2-36a$	$20a^2+48a$	$-8a^2-16a$
9	$-a^2-2a$	$5a^2+9a$	$-8a^2-12a$	$4a^2+5a$	$5a^2+10a$	$-25a^2-45a$	$40a^2+60a$	$-20a^2-25a$
10	$-a^2-4a-4$	$5a^2+19a+18$	$-8a^2-28a-24$	$4a^2+13a+10$	$4a^2+17a+18$	$-20a^2-81a-81$	$32a^2+120a+108$	$-16a^2-56a-45$
11	a^2+4a+4	$-5a^2-19a-18$	$8a^2+28a+24$	$-4a^2-13a-10$	$-5a^2-19a-18$	$25a^2+90a+81$	$-40a^2-132a-108$	$20a^2+61a+45$
12	a^2+2a	$-5a^2-9a$	$8a^2+12a$	$-4a^2-5a$	$-4a^2-8a$	$20a^2+36a$	$-32a^2-48a$	$16a^2+20a$
13	$-a^2$	$4a^2$	$-5a^2$	$2a^2$	$5a^2$	$-20a^2$	$25a^2$	$-10a^2$
14	$-a^2-2a$	$4a^2+8a$	$-5a^2-10a$	$2a^2+4a$	$4a^2+9a$	$-16a^2-36a$	$20a^2+45a$	$-8a^2-18a$
15	a^2+2a	$-4a^2-8a$	$5a^2+10a$	$-2a^2-4a$	$-5a^2-9a$	$20a^2+36a$	$-25a^2-45a$	$10a^2+18a$
16	a^2	$-4a^2$	$5a^2$	$-2a^2$	$-4a^2$	$16a^2$	$-20a^2$	$8a^2$

Column 9 through 16

Row								
1	$8a^2$	$-40a^2$	$64a^2$	$-32a^2$	$-4a^2$	$20a^2$	$-32a^2$	$16a^2$
2	$5a^2+12a$	$-25a^2-60a$	$40a^2+96a$	$-20a^2-48a$	$-2a^2-4a$	$10a^2+20a$	$-16a^2-32a$	$8a^2+16a$
3	$-8a^2-12a$	$40a^2+60a$	$-64a^2-96a$	$32a^2+48a$	$4a^2+5a$	$-20a^2-25a$	$32a^2+40a$	$-16a^2-20a$
4	$-5a^2$	$25a^2$	$-40a^2$	$20a^2$	$2a^2$	$-10a^2$	$16a^2$	$-8a^2$
5	$8a^2+16a$	$-32a^2-72a$	$40a^2+96a$	$-16a^2-32a$	$-4a^2-8a$	$16a^2+36a$	$-20a^2-48a$	$8a^2+16a$
6	$5a^2+22a+24$	$-20a^2-93a-108$	$25a^2+120a+144$	$-10a^2-44a-4$	$-2a^2-8a-8$	$8a^2+34a+36$	$-10a^2-44a-48$	$4a^2+16a+16$
7	$-8a^2-28a-24$	$32a^2+120a+108$	$-40a^2-156a-144$	$16a^2+56a+48$	$4a^2+13a+10$	$-16a^2-56a-45$	$20a^2+73a+60$	$-8a^2-26a-20$
8	$-5a^2-10a$	$20a^2+45a$	$-25a^2-60a$	$10a^2+20a$	$2a^2+4a$	$-8a^2-18a$	$10a^2+24a$	$-4a^2-8a$
9	$-8a^2-16a$	$40a^2+72a$	$-64a^2-96a$	$32a^2+40a$	$4a^2+8a$	$-20a^2-36a$	$32a^2+48a$	$-16a^2-20a$
10	$-5a^2-22a-24$	$25a^2+105a+108$	$-40a^2-156a-144$	$20a^2+73a+60$	$2a^2+8a+8$	$-10a^2-38a-36$	$16a2+56a+48$	$-8a^2-26a-20$
11	$8a^2+28a+24$	$-40a^2-132a-108$	$64a^2+192a+144$	$-32a^2-88a-60$	$-4a^2-13a-10$	$20a^2+61a+45$	$-32a^2-88a-60$	$16a^2+40a+25$
12	$5a^2+10a$	$-25a^2-45a$	$40a^2+60a$	$-20a^2-25a$	$-2a^2-4a$	$10a^2+18a$	$-16a^2-24a$	$8a^2+10a$
13	$-8a^2$	$32a^2$	$-40a^2$	$16a^2$	$4a^2$	$-16a^2$	$20a^2$	$-8a^2$
14	$-5a^2-12a$	$20a^2+48a$	$-25a^2-60a$	$10a^2+24a$	$2a^2+4a$	$-8a^2-16a$	$10a^2+20a$	$-4a^2-8a$
15	$8a^2+12a$	$-32a^2-48a$	$40a^2+60a$	$-16a^2-24a$	$-4a^2-5a$	$16a^2+20a$	$-20a^2-25a$	$8a^2+10a$
16	$5a^2$	$-20a^2$	$25a^2$	$-10a^2$	$-2a^2$	$8a^2$	$-10a^2$	$4a^2$

Note: See (4-13b) and Figure 4.3 for the order of parent pixels when using matrix $\boldsymbol{E}$.

Bibliography

[1] K. R. Castleman, *Digital Image Processing*, Prentice-Hall, 1996.

[2] Bernd Jähne, *Digital Image Processing—Concepts, Algorithms, and Scientific Applications*, 4th edition, Springer, 1997.

[3] W. K. Pratt, *Digital Image Processing*, New York: Wiley, 1991.

[4] A. K. Jain, Fundamentals of Digital Image Processing, Prentice-Hall, 1989.

[5] J.G.Proakis, *Digital Signal Processing—Principles, Algorithms, and Applications*, second edition, Macmillan.

[6] A. M. Tekalp, *Digital Video Processing*, Prentice Hall, 1995.

[7] J. A. Parker, R. V. Kenyon, and D. E. Troxel, "Comparison of Interpolating methods for Image Resampling", *IEEE Trans. on Image Processing*, Vol. MI-2, NO.1, March 1983.

[8] T. M. Lehmann, C. Gönner, and K. Spitzer, "Survey: Interpolation Methods in Medical Image Processing", *IEEE Trans. Medical Imaging*, Vol. 18, November 1999, pp.1049-1075.

[9] M. Unser, A. Aldroubi, and M. Eden, "Enlargement or Reduction of Digital Images with Minimum Loss of Information", *IEEE Trans. on Image Processing*, Vol.4, NO.3, March 1995.

[10] R. G. Keys, "Cubic Convolution Interpolation for Digital Image Processing," *IEEE Trans. Acoust., Speech, Signal Processing*, Vol. ASSP-29, NO. 6, pp.1153-1160.

[11] J. Allebach and P. W. Wong, "Edge-directed Interpolation", *Proceedings of ICIP'1996*, Vol. 3, Sept. 1996, pp.707-710.

[12] K. Jensen and D. Anastassiou, "Subpixel Edge Localization and the Interpolation of Still Images", *IEEE Trans. on Image Processing*, Vol.4, NO. 3, March 1995.

[13] S. Zahir, A. K. Murad Agha, and R. K. Ward, "A Near Exact Expansion for Bi-Level Images", *Proceedings of ICIP'2000*, pp.2627-2630.

[14] S. Carrato, G. Ramponi, and S. Marsi, "A Simple Edge-sensitive Image Interpolation Filter", *Proceeding of ICIP'1996*, pp711-714.

[15] X. Li and M. T. Orchard, "New Edge Directed Interpolation", *Proceeding of ICIP'2000*, pp.311-314.

[16] H, Jiang and C. Moloney, "A New Direction Adaptive Scheme for Image Interpolation," *Proceedings of ICIP'2002*, June 2002, Vol. 3, pp.369-372.

[17] Hongjian Shi and Rabab Ward, "Canny Edge Based Image Expansion", *Circuits and Systems, 2002. ISCAS 2002. IEEE International Symposium on*, Vol. 1, 2002, pp 785-788.

[18] J. W. Hwang and H. S. Lee, "Adaptive Image Interpolation Based on Local Gradient Features", *Signal Processing Letters, IEEE*, Vol. 11, Issue: 3, March 2004, pp. 359 – 362.

[19] C. R. Appledorn, "A new approach to the interpolation of sampled data," *Medical Imaging, IEEE Transactions on*, Vol. 15, Issue: 3, June 1996, pp.: 369 – 376.

[20] B. Morse and D. Schwartzwald, "Isophote-Based Interpolation", *Proceeding of ICIP'1998*, Vol.3, pp.227-231.

[21] B. Ayazifar and J. S. Lim, "Pel-Adaptive Model-Based Interpolation of Spatially Subsampled Images," *Proceedings of ICASSP-92*, Vol.3, March 1992, pp. 181-184.

[22] V. R. Algazi, G. E. Ford, and R. Potharlanka, "Directional Interpolation of Images Based on Visual Properties and Rank Order Filtering", *Proceeding of ICASSP'1991*, pp.3005-3008.

[23] P. Ferreira, "Two Fast Extrapolation/Superresolution Algorithms", *Proceeding of ICIP'2000*, Vol.2, pp343-346.

[24] B. S. Morse and D. Schwartzwald, "Image magnification using level-set Reconstruction," *Computer Vision and Pattern Recognition, Proceedings of the 2001 Computer Society Conference on*, Vol. 1, Dec. 2001, pp.333–340.

[25] Q. Wang and R. Ward, "A New Edge-Directed Image Expansion Scheme", *Proceeding of ICIP'2001*, Vol.3, pp.899-902.

[26] Q. Wang and R. Ward, "A Contour-Preserving Image Interpolation Method", *Proceedings of ICIP'2003*, Vol.3, pp.673-676.

[27] N. Dyn, D. Levin, and S. Rippa, "Data-Dependant Triangulation for Piecewise Linear Interpolation", *IMA J. Numerical Analysis*, Vol. 10, 1990, pp. 137 – 154.

[28] X. Yu, B. S. Morse and T. W. Sederberg, "Image Reconstruction Using Data-Dependent Triangulation, *Computer Graphics and Applications, IEEE*, Vol.21, Issue 3, May/Jun 2001, pp. 62–68.

[29] H. Aly and E. Dubois, "Regularized image up-sampling using a new observation model and the level set method", *Proceedings of ICIP'2003*, Vol. 3, Sept. 2003, pp. 665–668.

[30] C. Cai, T. H. Yu, and S. K. Mitra, "Saturation-based Adaptive Inverse Gradient Interpolation for Bayer Pattern Images", *Proc. IEE Vis. Image Signal Process.*, Vol. 148, June 2001, pp. 202 – 208.

[31] P. Thevenaz, T. Blu, and M. Unser, "Complete parameterization of piecewise polynomial interpolators according to degree support," *Proceedings of ICIP'2000*, Vol. 2, Sept. 2000, pp.335–338.

[32] N. A. Dodgson, "Quadratic Interpolation for Image Resampling," *IEEE Trans. on Image Processing*, Vol. 6, NO. 9, Sept. 1997.

[33] A. Schaum, "Theory and Design of Local Interpolators," *CVGIP: Graph. Models Image Processing*, Vol. 55, No. 6, 1993, pp.464 – 481.

[34] M. Unser, A. Aldroubi, and M. Eden, "Fast B-Spline Transforms for Continuous Image Representation and Interpolation", *IEEE Trans. on Pattern Analysis and Machine Intelligence*, Vol.13, NO.3, March 1991.

[35] E. Meijering and M. Unser, "A Note on Cubic Convolution Interpolation," *IEEE Trans. on Image Processing*, Vol.12, NO.4, April 2003.

[36] P. E. Danielsson and M. Hammerin, "Note: High accuracy rotation of images," *CVGIP: Graph. Model Image Processing*, Vol. 54, NO.4, 1992, pp.340-344.

[37] H. Theisel and G. Farin, "The Curvature of Characteristic Curves on Surfaces," *Computer Graphics and Applications, IEEE*, Vol. 17, Issue: 6, Nov.-Dec. 1997, pp. 88–96.

[38] M. Bentum, B. Lichtenbelt, and T. Malzbender, "Frequency Analysis of Gradient Estimators in Volume Rendering", *IEEE Trans. on Visualization and Computer Graphics*, Vol. 2, NO.3, September 1996, pp.242-254.

[39] Q. Wang, R. K. Ward, and H. Shi, "Isophote Estimation by Cubic-Spline Interpolation", *Proceedings of ICIP'2002*, Vol.3, June 2002, pp401-404.

[40] S. Osher, and J. A. Sethian, "Fronts Propagating with Curvature-Dependent Speed: Algorithms Based on Hamilton--Jacobi Formulations," *Journal of Computational Physics*, 79, 1988, pp. 12--49.

[41] H. T. Tanaka, O. Kling, and D. T. L. Lee, "On Surface Curvature Computation From Level Set Contours," *Proceedings of the 10th International Conference on Pattern Recognition*, Vol.1, June 1990, pp.155-160.

[42] J. A. Sethian, *Level Set Methods* — Evolving Interfaces in Geometry, Fluid Mechanics, Computer Vision, and Materials Science, Cambridge University Press, 1996.

[43] S. Osher and R. Fedkiw, *Level Set Methods and Dynamic Implicit Surfaces*, New York Springer, 2002.

[44] N. Damera-Venkata, T. D. Kite, W. S. Geisler, et al, "Image Quality Assessment Based on a Degradation Model", *IEEE Trans. on Image Processing*, Vol.9, NO.4, April 2000, pp.636-650.

[45] D. J. Heeger and P. C. Teo, "A Model of Perceptual Image Fidelity," *Proceedings of ICIP'1995*, Vol.2, Oct. 1995, pp.343-345.

[46] A. Huertas and G. Medioni, "Detection of Intensity Changes with Subpixel Accuracy Using Laplacian-Gaussian Masks," *IEEE Trans. on Pattern Analysis and Machine Intelligence*, Vol. PAMI-8, NO.5, Sept. 1986, pp.651-664.

[47] G. Pavlovic and A. M.Tekalp, "Maximum Likelihood Parametric Blur Identification Based on a Continuous Spatial Domain Model," *IEEE Trans. on Image Processing*, Vol.1, NO.4, Oct. 1992, pp.496-504.

[48] S. J. Reeves and M. Mersereau, "Blur Identification by the Method of Generalized Cross-Validation," *IEEE Trans. on Image Processing*, Vol.1, NO.3, July 1992, pp.301-311.

[49] O. K. Al-Shaykh and R. M. Mersereau, "Restoration of Lossy Compressed Noisy Images," *IEEE Trans. on Image Processing*, Vol.8, NO.10, Oct. 1999.

[50] R. L. de Queiroz, "Processing JPEG-Compressed Images and Documents," *IEEE Trans. on Image Processing*, Vol.7, NO.12, Dec. 1998.

[51] H. Greenspan, C. H. Anderson, and S. Akber, "Image Enhancement by Nonlinear Extrapolation in Frequency Space", *IEEE Trans. on Image Processing*, Vol.9, NO.6, June 2000.

[52] Y. L. Lee, H. C. Kin, and H. W. Park, "Blocking Effect Reduction of JPEG Images by Signal Adaptive filtering", *IEEE Trans. on Image Processing*, Vol.7, NO.2, February 1998.

[53] T. L. Ji, M. K. Sundareshan, and H. Roehrig, "Adaptive Image Contrast Enhancement Based on Human Visual Properties", *IEEE Trans. on Medical Imaging*, Vol.13, NO. 4, December 1994.

[54] T-C. Hsung, D. P-K. Lun, and W-C. Siu, "A Deblocking Technique for Block-Transform Using Wavelet Transform Modulus Maxima," *IEEE Trans. on Image Processing*, Vol.7, NO.10, Oct. 1998.

[55] A. Polesel, G. Ramponi, and V. J. Mathews, "Image Enhancement via Adaptive Unsharp Masking," *IEEE Trans. on Image Processing*, Vol.9, NO.3, March 2000.

[56] F. Russo, "An Image Enhancement Technique Combining Sharpening and noise Reduction," *IEEE Trans. on Instrumentation and Measurement*, Vol.51, No.4, August 2002.

[57] S. K. Mitra, H. Li, I.-S. Lin, and T-H Yu, "A New Class of Nonlinear Filters for Image Enhancement," *Proceeding of ICASSP-91*, Toronto, Canada, Apr. 1991, pp.2525-2528.

[58] A. Polesel, G. Ramponi, and V. J. Mathews, "Adaptive Unsharp Masking for Contrast Enhancement,", *Proceeding of ICIP'97*, Oct. 1997, Vol. 1, pp267-270.

[59] J. A. Stark, "Adaptive Image Contrast Enhancement Using Generalizations of Histogram Equalization," *IEEE Trans. on Image Processing*, Vol.9, NO.5, May 2000, pp.889-896.

[60] G. Birkhoff and S. MacLane, "*A Survey of Modern Algebra*", New York: Macmillan, 3rd edition, 1965.

[61] V. I. Smirnov, *A Course of Higher Mathematics, Volume I*, Translated by D.E.Brown, Pergamon Press, 1964.

[62] V. I. Smirnov, *A Course of Higher Mathematics, Volume II*, Translated by I.N.Sneddon, Pergamon Press, 1964.

[63] N. Merhav and V. Bhaskaran, "Fast Algorithms for DCT Domain Image Down-Sampling and for Inverse Motion Compensation", *IEEE Trans. on Circuits and Systems for Video Technology*, Vol.7, NO.3, June 1997, pp.468-476.

[64] E. Feig and S. Winograd, "Fast Algorithms for the Discrete Cosine Transform," *IEEE Trans. on Signal Processing*, Vol.40, NO.9, Sept. 1992, pp.2174-2193.

[65] C. Lee, M. Eden, and M. Unser, "High-Quality Image Resizing Using Oblique Projection operators," *IEEE Trans. on Image Processing*, Vol.7, NO.5, May 1998, pp.679-691.

[66] M. Tabei and M. Ueda, "Backprojection by Upsampled Fourier Series Expansion and Interpolated FFT," *IEEE Trans. on Image Processing*, Vol. 1, NO.1, Jan. 1992, pp.77-87.

[67] Y. Arai, T. Agui and M. Nakajima, "A Fast DCT-SQ Scheme for Images", *the Transactions of the IEICE*, Vol.E71, NO.11, November 1988, pp.1095-1097.

[68] F. Nicolier and F. Truchetet, "Image Magnification Using Decimated Orthogonal Wavelet Transform", *Proceedings of ICIP'2000*, Vol.2, Sept. 2000, pp. 355-358.

[69] S. A. Martucci, "Interpolation in the DST and DCT Domains," *Proceedings of ICIP'2000*, Vol.2, Sept. 2000, pp.339-342.

[70] R. Dugad and N. Ahuja, "A Fast Scheme for Downsampling and Upsampling in the DCT Domain," *Proceeding of ICIP'1999*, Vol.2, Oct. 1999, pp.909-913.

[71] N. Nguyen and P. Milanfar, "An Efficient Wavelet-based Algorithm for Image Superresolution," *Proceedings of ICIP'2000*, Vol.2, Sept. 2000, pp.351-354.

[72] K. Cinkler and A. Mertins, "Coding of Digital Video with the Edge-Sensitive Discrete Wavelet Transform," *Proceedings of ICIP'1996*, Vol.1, Sept. 1996, pp.961-964.

[73] D. H. Hubel and T. N. Wiesel, "Brain Mechanisms of Vision", SCI AM 241 (3): 150-&, 1979.

[74] D. H. Hubel and T. N. Wiesel, "Early Exploration of the Visual Cortex", NEURON 20 (3): 401-412, March 1998.

[75] J. Slater, *Modern Television Systems—to HDTV and Beyond*, Pitman Publishing, 1991.

[76] Alan Watt, and Fabio Policarpo, *3D Games—Real-time Rendering and Software Technology*, Volume One, Addison-Welsey, 2001.

[77] G. Wolberg, *Digital Image Warping*, Los Alamitos, CA, IEEE Comput. Soc. Press, 1990.

[78] Hongjian Shi, *Image Reduction and Edge-based Expansion*, Thesis, M.A.Sc, the University of British Columbia, 2002.

[79] A. K. Murad Agha, Image Expansion Using Segmentation-based Method, Thesis, M.A.Sc, the University of British Columbia, 2001.

[80] Recommendation ITU-T H.262 (1995 E).

[81] ITU-T Draft H.263 (1996).