

**ADAPTIVE BILATERAL EXTENSOR FOR
IMAGE INTERPOLATION**

A Thesis presented to the Faculty of the Graduate School
University of Missouri-Columbia

In Partial Fulfillment
of the Requirements for the Degree
of Master of Science

by

THOMAS J. RIEHLE JR.

Dr. Kannapan Palaniappan, Thesis Supervisor

MAY 2006

© copyright by Thomas J. Riehle Jr. 2006

All Rights Reserved

The undersigned, appointed by the Dean of the Graduate School, have examined the thesis entitled:

**ADAPTIVE BILATERAL EXTENSOR FOR
IMAGE INTERPOLATION**

presented by Thomas J. Riehle Jr.

a candidate for the degree of Master of Science

and hereby certify that in their opinion it is worthy of acceptance.

Dr. Kannapan Palaniappan

Dr. Jeffrey Uhlmann

Dr. David Larsen

Dr. Ye Duan

Acknowledgments

I extend my greatest appreciation to my advisor, Dr. Kannappan Palaniappan, for his constant support and guidance, without which this thesis could not be realized. I thank him for his countless hours, patience, and dedication to this thesis. I also thank Dr. Sumit Nath for his guidance in learning principles and applications necessary for conducting research within the field of computer science. His thoughts and ideas were always a meaningful contribution. I wish to acknowledge the support and assistance of Dr. Filiz Bunyak. Her willingness to assist in any way possible is greatly appreciated. I extend my gratitude to Dr. Jeffery Uhlmann. His willingness to listen and offer feedback has provided invaluable insight for this work. I also extend my sincere thanks to him for allowing me the pleasure to assist in the teaching of computer science 2050, algorithm design and programming II.

Finally, I dedicate this thesis to my wife, Nicole for her continued love and support throughout my graduate school career.

Contents

| | |
|---|-----------|
| Acknowledgements | ii |
| List of Figures | v |
| List of Tables | xi |
| 1 Introduction | 1 |
| 1.1 Background Information | 1 |
| 1.2 Problem Definition | 3 |
| 1.3 Justification of Proposed Research | 4 |
| 1.4 Proposed Scope of Research | 5 |
| 1.5 Thesis Organization | 5 |
| 2 Extensor Formulation | 7 |
| 2.1 Previous Work | 7 |
| 2.2 Isotropic Extensor-Based Formulation | 9 |
| 3 Two-Dimensional Structure Tensor Based Orientation Estimation | 13 |
| 3.1 Two Dimensional Structure Tensor Based Edge and Orientation Esti- mation | 13 |
| 4 Adaptive Bilateral Extensor (ABE) Interpolation | 19 |

| | | |
|----------|---|-----------|
| 4.1 | Bilateral Filtering | 19 |
| 4.2 | Bilateral Interpolation Algorithm | 24 |
| 4.3 | Bilateral Interpolation Results | 25 |
| 4.4 | Adaptive Bilateral Extensor Interpolation Algorithm | 26 |
| 4.4.1 | Adaptive Bilateral Extensor Algorithm | 33 |
| 5 | Performance Evaluation Metrics and Results | 35 |
| 5.1 | Image Evaluation Metrics | 35 |
| 5.1.1 | Structural Similarity Quality Evaluation | 37 |
| 5.2 | Interpolation Results | 39 |
| 5.3 | Effect of Regularization | 61 |
| 6 | Pan Sharpening for Multi-Resolution Image Data Sets | 63 |
| 6.1 | Multi-Resolution Remote Sensing Imagery | 63 |
| 6.2 | Pan-Sharpener Results | 69 |
| 7 | Conclusion | 83 |
| 7.1 | Contributions | 83 |
| 7.2 | Future Work | 84 |
| A | Adaptive Bilateral Extensor (ABE) Implementation | 85 |
| | References | 88 |

List of Figures

- 2.1 Pixel indexes with respect to the current pixel (i, j) (left). $\vec{\mathbf{X}}_i$ for each pixel i in the extensor window (middle) where i is an index based on lexicographic ordering of the pixels in the window. Distance calculation for one row of \mathbf{X} for an isotropic extensor window containing nine positions (right). In this figure, $\vec{\varphi}_1^T = [\varphi_{11}, \varphi_{12}, \dots, \varphi_{1N}] = [0, 1, 2, 1, \sqrt{2}, \sqrt{3}, 2, \sqrt{3}, 2\sqrt{2}]$ 11
- 2.2 $\vec{\varphi}_y$ calculated based on values within the extensor window. The sample distance calculation is from location $\vec{\mathbf{X}}_i$ to $\vec{\mathbf{y}}$, the pixel position to interpolate. 12
- 2.3 Calculation of φ_i . The i^{th} row of extensor matrix \mathbf{X} , which are the distances from $\vec{\mathbf{X}}_1$ to each of the pixel locations within the extensor window. Note that the origin of every vector (i.e. distance calculation) is the same position $\vec{\mathbf{X}}_1$ 12
- 3.1 Gradient and edge orientations of a pixel located in an ideal edge . . 14
- 3.2 The first iterations of the spatially adaptive Gaussian window algorithm used in the adaptive structure tensor computation. R_i^1, R_i^2 are the semi-major and minor axes, respectively, at iteration i and Ω_i is the corresponding neighborhood. 17

| | | |
|------|--|----|
| 4.1 | Comparison of (left to right) Gaussian, Tukey, and Geman-McClure kernel functions | 21 |
| 4.2 | 512x512 Lena Sjöoblom with areas utilized for bilateral filtering. | 22 |
| 4.3 | Comparison of (top to bottom) Gaussian, Tukey, and Geman McClure kernels for bilateral filtering with $\sigma_d = 10$ and $\sigma_i = 10, 30, 100$ in each of the columns (left to right) respectively. | 23 |
| 4.4 | Depiction of original pixels, denoted in black, in the interpolated result. | 24 |
| 4.5 | Computation of bilateral weights for I_{xy} | 24 |
| 4.6 | A 128x128 portion of Lena processed using bilateral interpolation (without extensor component) by a factor of two with $\sigma_i = 90$ | 25 |
| 4.7 | Hat brim region of the original Lena image interpolated by a factor of 2 using bilateral interpolation (left) with $\sigma_i = 120$, $\sigma_d = 10$ and bicubic interpolation (right) | 26 |
| 4.8 | Mask \vec{m} based on included pixels (in black), excluded pixels (patterned), pixels to interpolate (in white). The pixel to be interpolated is identified as $I_z(y)$ and is within the group of black labeled pixels in this example. | 27 |
| 4.9 | Pixels included for interpolation using adaptive bilateral extensor | 32 |
| 4.10 | Pixels included in regularization of x_{ij} | 32 |
| 4.11 | Diagram of Adaptive Bilateral Extensor Interpolation algorithm. | 32 |
| 4.12 | Estimate of edge at iteration I^k | 33 |
| 4.13 | Estimate of edge at iteration $I^{(k+1)}$ | 33 |
| 5.1 | Depiction of decimation employed before interpolation. The original image with rows and columns marked for removal (left). Resulting decimated image (right). | 40 |
| 5.2 | Original house image with regions of interest as labeled. | 41 |

| | | |
|------|--|----|
| 5.3 | Decimated tire region of house image. | 41 |
| 5.4 | Original tire region of house image. | 41 |
| 5.5 | Bicubic interpolation by factor two. | 41 |
| 5.6 | IE interpolation by factor two. | 41 |
| 5.7 | Edge-Directed interpolation by factor two. | 42 |
| 5.8 | ABE interpolation by factor two. | 42 |
| 5.9 | Decimated leaves region. | 42 |
| 5.10 | Original leaves region. | 42 |
| 5.11 | Bicubic interpolation by factor four. | 44 |
| 5.12 | IE interpolation by factor four. | 44 |
| 5.13 | Edge-directed interpolation by factor four. | 45 |
| 5.14 | ABE interpolation by factor four. | 45 |
| 5.15 | Decimated trees region. | 46 |
| 5.16 | Original trees region. | 46 |
| 5.17 | Bicubic interpolation by factor eight. | 46 |
| 5.18 | IE interpolation by factor eight. | 47 |
| 5.19 | Edge-Directed interpolation by factor eight. | 48 |
| 5.20 | ABE interpolation by factor eight. | 49 |
| 5.21 | Original airplane (F-16) image with region of interest as labeled. . . | 50 |
| 5.22 | Decimated airplane (F-16) image. | 50 |
| 5.23 | Original airplane (F-16) image. | 50 |
| 5.24 | Bicubic interpolation by factor eight. | 51 |
| 5.25 | IE interpolation by factor eight. | 51 |
| 5.26 | Edge-Directed interpolation by factor eight. | 52 |
| 5.27 | ABE interpolation by factor eight. | 52 |
| 5.28 | Decimated red brick house image. | 53 |

| | | |
|------|---|----|
| 5.29 | Original sampling of red brick house image. | 53 |
| 5.30 | Decimated image of pipe region. | 53 |
| 5.31 | Original image of pipe region. | 53 |
| 5.32 | Bicubic interpolation of pipe region by factor eight. | 54 |
| 5.33 | IE interpolation of pipe region by factor eight. | 54 |
| 5.34 | Edge-Directed image of pipe region at factor eight. | 54 |
| 5.35 | ABE Interpolation of pipe region at factor eight. | 54 |
| 5.36 | Region of remote sensing image at 2.4 meters per pixel. | 55 |
| 5.37 | Original house region. | 56 |
| 5.38 | Original pools region. | 56 |
| 5.39 | Bicubic interpolation by factor four. | 56 |
| 5.40 | IE interpolation by factor four. | 56 |
| 5.41 | Edge-directed image by factor four. | 56 |
| 5.42 | ABE interpolation by factor four. | 56 |
| 5.43 | Bicubic interpolation by factor eight. | 57 |
| 5.44 | IE Interpolation by factor eight. | 57 |
| 5.45 | Edge-directed image by factor eight. | 58 |
| 5.46 | ABE interpolation by factor eight. | 58 |
| 5.47 | Region of decimated image for regularization. | 61 |
| 5.48 | Region of original image for regularization. | 61 |
| 5.49 | Interpolation factor two with regularization. | 61 |
| 5.50 | Interpolation factor two without regularization. | 61 |
| 5.51 | Interpolation by zoom factor two with regularization, nearest neighbor interpolation by additional factor two. | 62 |
| 5.52 | Interpolation by zoom factor two, without regularization, nearest neighbor interpolation by additional factor two. | 62 |

| | | |
|------|---|----|
| 5.53 | Interpolation by factor four with regularization, nearest neighbor up-sampling by additional factor four. | 62 |
| 5.54 | Interpolation by factor four without regularization, nearest neighbor upsampling by additional factor four. | 62 |
| 6.1 | Red channel of the remote sensing region. | 64 |
| 6.2 | Green channel of the remote sensing region. | 65 |
| 6.3 | Blue channel of the remote sensing region. | 65 |
| 6.4 | Near Infrared (IR) channel of the remote sensing region. | 66 |
| 6.5 | Combined RGB channels of remote sensing region. | 66 |
| 6.6 | Scaled image of the panchromatic channel of the remote sensing region. | 67 |
| 6.7 | Pan-sharpening algorithm. | 68 |
| 6.8 | Region of remote sensing image at 2.4m resolution with regions of interest as labeled. The image is scaled for display, smaller than the original size image. | 70 |
| 6.9 | Region of 0.6m Pan image with regions further utilized as labeled, scaled for display corresponding to the 2.4m data. | 70 |
| 6.10 | Original region of the orchard in the MS image | 70 |
| 6.11 | Bicubic interpolation of the orchard region at zoom factor four. | 71 |
| 6.12 | IE interpolation of the orchard region at zoom factor four. | 71 |
| 6.13 | Edge-directed interpolation of the orchard region at zoom factor four. | 72 |
| 6.14 | ABE interpolation of the orchard region at zoom factor four. | 72 |
| 6.15 | Original region of the buildings in the MS image | 73 |
| 6.16 | Bicubic interpolation of the buildings region at zoom factor four. | 73 |
| 6.17 | IE interpolation of the buildings region at zoom factor four. | 74 |
| 6.18 | Edge-directed interpolation of the buildings region at zoom factor four. | 74 |
| 6.19 | ABE interpolation of the buildings region at zoom factor four. | 75 |

| | | |
|------|--|----|
| 6.20 | Original region of the orchard in the Pan image | 77 |
| 6.21 | Pan-sharpening of orchard region using bicubic interpolation for up- sampling the MS image. | 78 |
| 6.22 | Pan-sharpening of orchard region using IE algorithm for upsampling the MS image. | 78 |
| 6.23 | Pan-sharpening of orchard region using edge-directed interpolation for upsampling the MS image. | 79 |
| 6.24 | Pan-sharpening of orchard region using the ABE algorithm for upsam- pling the MS image. | 79 |
| 6.25 | Original region of the orchard in the Pan image | 80 |
| 6.26 | Pan-sharpening of buildings region using bicubic interpolation for up- sampling the MS image. | 80 |
| 6.27 | Pan-sharpening of buildings region using IE algorithm for upsampling the MS image. | 81 |
| 6.28 | Pan-sharpening of buildings region using edge-directed interpolation for upsampling the MS image. | 81 |
| 6.29 | Pan-sharpening of buildings region using the ABE algorithm for up- sampling the MS image. | 82 |

List of Tables

| | | |
|-----|--|----|
| 1.1 | Specifications of the QuickBird satellite | 2 |
| 1.2 | Specifications of the IKONOS satellite | 3 |
| 5.1 | Parameter values used for extensor-based interpolation methods . . . | 40 |
| 5.2 | Comparison of SSIM for selected interpolation algorithms. The decimated image is interpolated by a factor of two and numerically compared to the original image as a percent of similarity. | 59 |
| 5.3 | Comparison of SNR for selected interpolation algorithms. The decimated image is upsampled by a factor of two and compared to the original image. | 60 |
| 6.1 | Comparison of SNR for selected interpolation algorithms using the remote sensing images. The image is decimated by a factor of two, then upsampled by a factor of two and compared to the original image. | 76 |
| 6.2 | Comparison of SSIM for selected interpolation algorithms using the remote sensing images. The image is decimated by a factor of two, then upsampled by a factor of two and compared to the original image. | 76 |

Chapter 1

Introduction

1.1 Background Information

THE digital revolution at the turn of the 21st century has dramatically changed society. As a result, technology driven devices are employed everywhere. A common feature in these devices is the ability to capture a digital image. Remote sensing satellites capture images throughout the world for both commercial and defense applications. Mobile phones and digital cameras allow one to quickly and easily obtain a photo. Charge-coupled devices (CCD) are also embedded in numerous additional devices [1]. The immediate availability of digital images has in turn created demand for software and applications that can readily utilize and process these images.

Standalone digital cameras are commonplace and capture high quality photos. However, the storage space and bandwidth requirements can still be cost prohibitive. Images are often compressed to reduce transmission and storage, but quality is degraded as a result of the compression process. Most mobile phones are only capable of capturing images at a very low resolution. It is desirable to obtain a method by which the quality of an image can be restored after compression and transmission.

The resolution capabilities of digital imaging devices are continually improving [2, 3]. Display devices to view these images are also capable of higher resolutions. However, there are large archives of digital images captured at significantly lower resolutions. It is desirable to improve the resolution of these images to leverage the advantages of current state-of-the-art applications. Image interpolation, also known as upsampling, aims to develop techniques to supersample these images to a larger resolution while maintaining the detail present in the original image.

Satellite acquired remote sensing imagery is an increasingly important resource for scientific, commercial, and military applications. Current satellites are equipped with several CCDs capable of capturing images at multiple resolutions and in different spectral wavelengths [4]. High resolution multispectral images include panchromatic (Pan), where wavelengths from blue through near-infrared (IR) are captured as a single channel of data resulting in a detail rich grayscale image with spatial resolution of 0.6m for QuickBird [5]. Also available are lower resolution images that contain four channels of data representing red, green, blue (RGB), and IR that are displayed together in a multi-spectral (MS) image. The majority of applications prefer a color image representation of remote sensing data, but also desire the more detailed information present in the high resolution Pan image. Table 1.1 shows the capabilities of the QuickBird remote sensing satellite [6]. The specifications of the IKONOS satellite [7] are contained in Table 1.1.

Table 1.1: Specifications of the QuickBird satellite

| Band | Spectral Wavelength (nm) | Spatial Resolution (m) |
|-------|--------------------------|------------------------|
| Blue | 450 - 520 | 2.44 |
| Green | 520 - 600 | 2.44 |
| Red | 630 - 690 | 2.44 |
| IR | 760 - 900 | 2.44 |
| Pan | 450 - 900 | 0.61 |

Table 1.2: Specifications of the IKONOS satellite

| Band | Spectral Wavelength (nm) | Spatial Resolution (m) |
|-------|--------------------------|------------------------|
| Blue | 444.7 - 516.0 | 1.0, 4.0 |
| Green | 506.4 - 595.0 | 1.0, 4.0 |
| Red | 631.9 - 697.7 | 1.0, 4.0 |
| IR | 757.3 - 852.7 | 1.0, 4.0 |
| Pan | 525.8 - 928.5 | 1.0 |

1.2 Problem Definition

Image interpolation, or image zooming, as presented in this thesis is formulated as follows. Consider an input image I with dimensions 256x256. The resolution of the image may be insufficient, perhaps due to compression, decimation, or a low resolution CCD initially used for image capture [1]. Performing interpolation on I will increase the dimensions by a specified *zoom factor*. For example, performing interpolation by a zoom factor of two on a 256x256 image will increase the dimensions of the resulting image I' to 512x512. It is common to perform interpolation using integer factors such as two or four.

Multispectral data fusion of RGB and Pan remote sensing data is referred to as *pan-sharpening*. For example, sensors onboard Quickbird, among other satellites, capture both 2.4 meters per pixel MS and 0.6 meter Pan imagery. While the Pan image provides a superior resolution and range of data, additional detail can be leveraged by combining the MS and Pan imagery [8]. Moreover, imagery presented in full color is desirable for consumer applications and publishing. Pan-sharpening is used to enhance lower resolution MS imagery by combining channels of varying resolution [9]. It combines the channels of color present in a MS image with the detail and high resolution of a Pan image to produce an image that retains the detail of the Pan image while employing the MS channels for a color fused result [10].

1.3 Justification of Proposed Research

Current methods for image interpolation initially appear satisfactory. Edge-adaptive zooming interpolates pixel values based on a set of criterion taken from detecting edges and discontinuities, while maintaining computational complexity below that of bicubic interpolation [11]. However, closer inspection reveals the introduction of structural artifacts that can add new details near edge boundaries and ringing artifacts near circular and oval regions of the resulting image [12]. Artifacts are clearly undesirable. An interpolation method should not introduce new details or information, which therefore compromises the authenticity of the image. Maintaining authenticity is especially important for scientific, military, and medical applications where image detail drives critical decision making.

Classic interpolation methods including bilinear and bicubic interpolation [13] are computationally efficient and simple to implement. However, these methods suffer from aliasing along and near edge boundaries. Areas of high contrast appear to suffer the most noticeable aliasing effects resulting in an unsatisfactory result in most practical applications. Attempting to supersample an image to a high resolution aims to minimize aliasing, blurring, and introduction of artifacts.

An important step in pan-sharpening is accurately upsampling the images. Without accurate interpolation, the fused data will result in *color bleeding* beyond their original boundaries. It is desirable to enhance features of the Pan image without losing detail. Bicubic interpolation is commonly employed to upsample the MS image, yielding a reasonable result. However, color bleeding and aliasing are still significant concerns.

1.4 Proposed Scope of Research

Adaptive Bilateral Extensor (ABE) interpolation is introduced in this thesis and leverages the following applications:

- ABE image interpolation applied to a decimated or subsampled target image resulting in a superior super-resolution image compared to current state-of-the-art methods.
- Bilateral Interpolation for lossless quality original images.
- ABE interpolation for pan-sharpening of remote sensing imagery.

1.5 Thesis Organization

The organization of this thesis is outlined as follows. Chapter 1 begins with providing background information and motivation for the research proposed in this thesis. A formal presentation of the problems addressed are next explained. The introductory chapter is concluded with justification for the research proposed in this work.

Current and classic algorithms employed for image interpolation are discussed at the beginning of Chapter 2. The extensor formulation is next presented, providing a detailed mathematical foundation for the extensor-based interpolation algorithm. Chapter 3 follows by introducing the mathematical formulation of the two dimensional structure tensor employed for ABE interpolation.

Chapter 4 begins with an introduction to bilateral filtering. It follows by formulating a novel adaptive bilateral image interpolation algorithm. The ABE interpolation algorithm is then presented as a combination of extensor-based interpolation and bilateral filtering.

A discussion of desirable properties for image quality evaluation begin Chapter 5. The metrics chosen for performance evaluation of image interpolation algorithms are then introduced. Results obtained for image upsampling are next presented. It follows with analysis and discussion of results compared to current state-of-the-art interpolation techniques. Qualitative and quantitative comparisons are presented and discussed with respect to current interpolation algorithms.

Preliminary information on satellite remote sensing data begins Chapter 6. It continues with the ABE interpolation algorithm for the application of pan-sharpening. Results and comparison to other interpolation algorithms for pan-fusion are next presented. This thesis concludes with a summary of accomplishments and dialogue for future work in Chapter 7.

Chapter 2

Extensor Formulation

Overview

This chapter begins with an overview of classic image interpolation algorithms and then introduces the isotropic extensor algorithm developed by Palaniappan et. al. [12] and Li [14]. Low order interpolants are first discussed. The recently developed edge-directed interpolation method by Orchard and Li [15] is also described as it will be used as a comparison method for benchmarking. The mathematical formulation and background information regarding the extensor-based method is described as it will be extended to the proposed adaptive bilateral extensor in Chapter 4.

2.1 Previous Work

NUMEROUS algorithms to solve the issue of image interpolation have been proposed in the past several decades. A simple naive method is nearest neighbor interpolation. Nearest neighbor interpolation proceeds by selecting as the interpolated value the pixel with the smallest spatial distance to the current pixel position in the interpolated domain. One advantage of this process is that the color table of the image remains intact. This property is very desirable for image editing

software when examining an image at high zoom factors. While computation of this algorithm is efficient, nearest neighbor interpolation produces blocky structures and significant aliasing along edge boundaries.

A related method more widely used is bilinear interpolation which utilizes the four nearest pixels to a point (x, y) to be interpolated. By formulating four equations and solving for the unknown variables, we can obtain coefficients $a \cdots d$ and use

$$I'_{xy} = ax + by + cxy + d \quad (2.1)$$

to obtain the pixel intensity $I'_{(x,y)}$ [16]. Bilinear interpolation maintains computational efficiency and improves over that of nearest neighbor interpolation, though aliasing along edge boundaries still persists. Perhaps the most well known classic interpolation method is bicubic interpolation formulated as [13].

$$I'_{x'y'} = \sum_{i=0}^3 \sum_{j=0}^3 D_{ij} x^i y^j \quad (2.2)$$

where D_{ij} represents the respective distances between pixels. Bicubic interpolation is one of the best methods currently available for image interpolation producing well defined edges with minimal aliasing. However, the computational complexity of this method is high, requiring significant execution time to generate the resulting image.

Several methods have been proposed recently that improve on classic interpolation methods. Cubic convolution improves on classic interpolation methods, but requires computationally expensive operations to implement [17]. Edge-directed interpolation introduced adaptation based on covariance information, maintaining a correspondence between the low-resolution original image and the high-resolution interpolated result [15]. A data-adaptive method is then employed with covariance-based adaptive and bilinear interpolation. While producing a visually pleasing result, a closer inspection

reveals that edge-directed interpolation introduces new features and swirling artifacts that are not present in the original image. A method proposed in [18] employs a gradient search algorithm to enhance performance near edge boundaries. An isotropic extensor-based (IE) method utilizing a nonlinear mapping between the spatial distance and pixel intensity produces results better than that of bilinear interpolation while preserving texture and underlying information [12]. However, it is unable to completely overcome aliasing along edge boundaries.

Classic image interpolation methods produce a reasonable approximation to a supersampled result. However, these linear methods cannot prevent aliasing along edge boundaries. Nearest neighbor and bilinear interpolation especially suffer from this undesirable effect. Recent methods attempt to minimize aliasing along edges by employing data adaptive methods. Unfortunately, these often introduce artifacts and features that are not present in the original image.

2.2 Isotropic Extensor-Based Formulation

An isotropic extensor-based algorithm is presented in this thesis based on the previous work in [12, 14]. Its foundation is rooted in the hypothesis that a transformation matrix T can be found to express a mapping that exists between pixel intensities and their euclidean distance. A supersampled image I' is obtained from a decimated image I by utilizing the mapping between intensities and their lifted-distances. A function which can raise a d -dimensional vector into a superspace of N dimensions is hereafter known as an *extensor function*. An N -dimensional matrix consisting of N extensor functions will be defined as an *extensor matrix*, where d represents the dimension of the data. We consider d as two dimensions in this thesis since we are working with digital images. Certain mathematical properties are preferable to possess for extensor functions. One such property common to many types of transformations is

$f(Lx) = Lf(x)$ given a linear transformation L and extensor function f . It also may be desirable to have a similar property hold for restricted transformations such as rotation and scaling. Additional classes of extensor functions may also be satisfactory to a visual inspection.

The class of extensor functions utilized in this thesis are defined based on the euclidean distance between pairs of pixel index values within a specified area known as an *extensor window*. It is often an isotropic area, formulated as

$$\mathbf{X} = \frac{1}{K} \begin{bmatrix} \varphi_{11} & \varphi_{12} & \cdots & \varphi_{1N} \\ \varphi_{21} & \varphi_{22} & \cdots & \varphi_{2N} \\ \vdots & \vdots & \vdots & \vdots \\ \varphi_{N1} & \varphi_{N2} & \cdots & \varphi_{NN} \end{bmatrix}_{N \times N} \quad (2.3)$$

where

$$\varphi_{ij} = h(\vec{X}_i - \vec{X}_j) = \left\| \vec{X}_i - \vec{X}_j \right\|, \quad i, j = 1, 2, \dots, N \quad (2.4)$$

and $\left\| \vec{X}_i - \vec{X}_j \right\|$ denotes the Euclidean distance and X_1, X_2, \dots, X_N are a lexicographical ordering of the pixel positions between pixel position \vec{X}_i and \vec{X}_j ; with both being within the extensor window; K denotes the normalization term based on the desired zoom factor.

Let ϕ_y denote a single extensor at pixel y in Eq 2.3 formulated as

$$\varphi_y = Ext(\vec{y}) = \begin{bmatrix} \left\| \vec{X}_1 - \vec{y} \right\| \\ \left\| \vec{X}_2 - \vec{y} \right\| \\ \vdots \\ \left\| \vec{X}_N - \vec{y} \right\| \end{bmatrix} \quad (2.5)$$

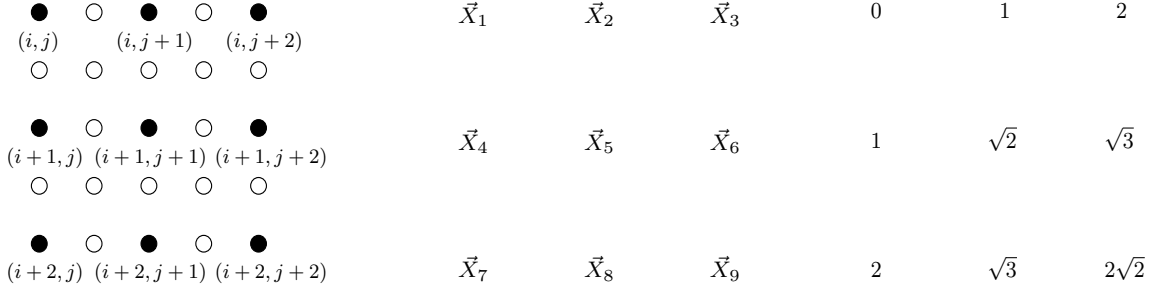


Fig. 2.1: Pixel indexes with respect to the current pixel (i, j) (left). \vec{X}_i for each pixel i in the extensor window (middle) where i is an index based on lexicographic ordering of the pixels in the window. Distance calculation for one row of \mathbf{X} for an isotropic extensor window containing nine positions (right). In this figure, $\vec{\varphi}_1^T = [\varphi_{11}, \varphi_{12}, \dots, \varphi_{1N}] = [0, 1, 2, 1, \sqrt{2}, \sqrt{3}, 2, \sqrt{3}, 2\sqrt{2}]$

The intensity domain is denoted by $\tilde{\mathbf{G}}$, corresponding to g intensity dimensions, where $g=1$ for grayscale and $g=3$ for RGB color images, holds the current intensity values within the extensor window. It is employed to solve for the pixel intensity transformation matrix T by

$$\mathbf{T}_{3 \times N} \mathbf{X}_{N \times N} = \tilde{\mathbf{G}}_{3 \times N} \quad (2.6)$$

Solving for the transformation matrix T will reveal the mapping to interpolate the intensity value at a specified index as

$$\mathbf{T}_{3 \times N} = \tilde{\mathbf{G}}_{3 \times N} \mathbf{X}_{N \times N}^{-1} \quad (2.7)$$

Utilizing Eq 2.3 and Eq 2.4 we are assured a distance matrix that is positive definite, and henceforth it is invertible [11]. Correcting the intensity for constant regions

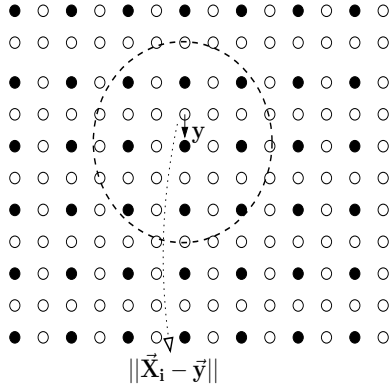


Fig. 2.2: $\vec{\varphi}_y$ calculated based on values within the extensor window. The sample distance calculation is from location \vec{X}_i to \vec{y} , the pixel position to interpolate.

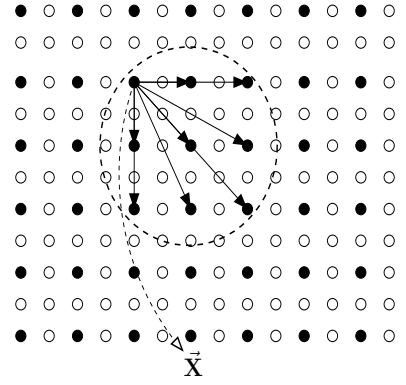


Fig. 2.3: Calculation of φ_i . The i^{th} row of extensor matrix \mathbf{X} , which are the distances from \vec{X}_1 to each of the pixel locations within the extensor window. Note that the origin of every vector (i.e. distance calculation) is the same position \vec{X}_1 .

within the extensor window ensures a uniform interpolation throughout, leading to

$$\tilde{\mathbf{G}} = \begin{bmatrix} G_{1,1} - \mu_1 & G_{1,2} - \mu_1 & \cdots & G_{1,N} - \mu_1 \\ G_{2,1} - \mu_2 & G_{2,2} - \mu_2 & \cdots & G_{2,N} - \mu_2 \\ G_{3,1} - \mu_3 & G_{3,2} - \mu_3 & \cdots & G_{3,N} - \mu_3 \end{bmatrix}_{3 \times N}, \quad \vec{\mu} = \begin{bmatrix} \mu_1 \\ \mu_2 \\ \mu_3 \end{bmatrix} \quad (2.8)$$

where $\mu = [\mu_1 \ \mu_2 \ \mu_3]^T$ is the local mean within the interpolation window. Suitable representations for the correction factor include the mean or median value within the extensor window. Finally, we can solve for the interpolated pixel intensity values by

$$I_z y = \mathbf{T} \vec{\varphi}_y = \tilde{\mathbf{G}} \mathbf{X}^{-1} \varphi_y + \vec{\mu} \quad (2.9)$$

where $I_z y$ denotes the interpolated value in the resulting image at pixel location \vec{y} with extensor representation φ_y .

Chapter 3

Two-Dimensional Structure Tensor Based Orientation Estimation

Overview

The first implement to the isotropic structure tensor is to incorporate information about the local image structure, particularly edges. The adaptive structure tensor, based on the paper by Nath and Palaniappan [19], is utilized in the adaptive bilateral extensor interpolation algorithm as introduced in Chapter 4 is then presented.

3.1 Two Dimensional Structure Tensor Based Edge and Orientation Estimation

EDGE and orientation estimation is performed utilizing the adaptive structure tensor algorithm. The summary of the adaptive structure tensor algorithm in this chapter is based on the paper by Nath and Palaniappan [19]. Consider a small region of an image, denoted as $\Omega(y)$, with center location x and current location y . The actual gradient of $\Omega(y)$ is denoted by $\mathbf{v}(x)$. The error vector between the

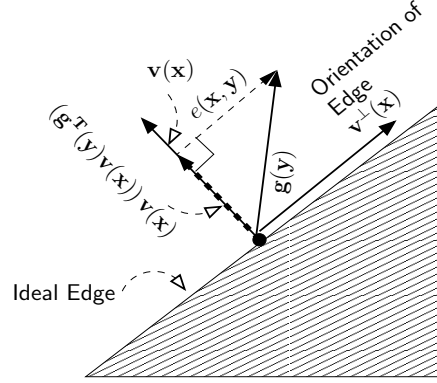


Fig. 3.1: Gradient and edge orientations of a pixel located in an ideal edge

estimated and actual gradients is expressed by

$$\|\mathbf{e}(x, y)\| = \|\mathbf{g}(y) - (\mathbf{g}^T(y), \mathbf{v}(x)) \mathbf{v}(x)\| \quad (3.1)$$

An illustration of the error is provided in Figure 3.1. To obtain the best possible estimate of $\mathbf{v}(x)$, $\rho(\|\mathbf{e}(x, y)\|^2)$ is minimized by integrating over Ω such that the magnitude of \mathbf{v} and \mathbf{g} is of unit length. Hereafter referred to as the *least-squares* error functional, the total error over Ω is expressed as

$$e_{LS}(\mathbf{x}) = \int_{\Omega} \rho(\|\mathbf{e}(x, y)\|^2) W(x, y) dy \quad (3.2)$$

where $W(x, y)$ is a Gaussian weighting function that enhances the gradient when evaluating the structure tensor with respect to the center pixel inside Ω and $\rho(\|\mathbf{e}(x, y)\|) = \|\mathbf{e}(x, y)\|^2$. This equation can then be simplified into two parts as

$$e_{LS}(\mathbf{x}) = \int_{\Omega} (\mathbf{g}^T \mathbf{g}) W(\mathbf{x}, \mathbf{y}) dy - \int_{\Omega} (\mathbf{v}^T (\mathbf{g} \mathbf{g}^T) \mathbf{v}) W(\mathbf{x}, \mathbf{y}) dy \quad (3.3)$$

The minimization of Eq 3.3 can also be turned into a maximization problem. If the constraint $\|\mathbf{v}\| = 1$ is applied, then maximizing the second term of Eq 3.3 can be

solved using Lagrange multipliers as

$$\mathcal{E}_{LS}(\mathbf{x}, \mathbf{y}) = \mathbf{v}^T \left(\int_{\Omega} (\mathbf{g}\mathbf{g}^T) \mathbf{W}(x, y) d\mathbf{y} \right) \mathbf{v} + \lambda(1 - \mathbf{v}^T \mathbf{v}) \quad (3.4)$$

$e_{LS}(x, y)$ can now be differentiated to find the boundaries of the function. This is equivalent to the standard eigenvalue problem and solving for \mathbf{v} to obtain the best possible estimate, $\hat{\mathbf{v}}$.

$$\mathbf{J}(\mathbf{x}, W) \hat{\mathbf{v}} = \lambda \hat{\mathbf{v}} \quad (3.5)$$

where

$$\mathbf{J}(\mathbf{x}, W) = \int_{\Omega} (\mathbf{g}\mathbf{g}^T) W(\mathbf{x}, \mathbf{y}) d\mathbf{y}$$

represents the least-squares structure tensor with weighting kernel W at position x . The least squares estimate of the gradient at x employing the available information in Ω is obtained via the maximum eigenvector in 3.5. Additionally, the Geman-McClure robust function [20]

$$\rho(\|\mathbf{e}(\mathbf{x}, \mathbf{y})\|, m) = \frac{\|\mathbf{e}(\mathbf{x}, \mathbf{y})\|^2}{m^2 + \|\mathbf{e}(\mathbf{x}, \mathbf{y})\|^2} = 1 - \frac{m^2}{m^2 + \|\mathbf{e}(\mathbf{x}, \mathbf{y})\|^2} \quad (3.6)$$

is utilized as it is more robust to outliers and m is a tuning parameter for devaluing outlying data. The error function employing a Geman-McClure robust function can be presented from 3.6 as

$$e_{GM} = \int_{\Omega} W(\mathbf{x}, \mathbf{y}) d\mathbf{y} - \int_{\Omega} \frac{m^2}{(\mathbf{g}^T \mathbf{g} - \mathbf{v}^T (\mathbf{g}\mathbf{g}^T) \mathbf{v} + m^2)} W(\mathbf{x}, \mathbf{y}) d\mathbf{y} \quad (3.7)$$

Assuming the same constraints as discussed above for the Gaussian function and again employing Lagrange multipliers and differentiating as above results in the Geman-

McClure robust tensor function.

$$\mathbf{J}(\mathbf{x}, \mathbf{v}, W) = \int_{\Omega} \frac{m^2}{(\mathbf{g}^T \mathbf{g} - \mathbf{v}^T (\mathbf{g} \mathbf{g}^T) \mathbf{v} + m^2)^2} (\mathbf{g} \mathbf{g}^T) W(\mathbf{x}, \mathbf{y}) d\mathbf{y} \quad (3.8)$$

An iterative process can then be utilized to converge 3.8 to a local minimum within Ω as follows.

$$\mathbf{J}(\mathbf{x}, \mathbf{v}_i, W) \mathbf{v}_{i+1} = \lambda \mathbf{v}_{i+1} \quad (3.9)$$

The above discussion utilizes a fixed size Gaussian window to estimate edge orientation. As evidenced in [19], a spatially varying window provides a more accurate estimate of orientation and is utilized in this thesis. An isotropic Gaussian window is used initially and then adapted based on size and orientation to refine the edge orientation estimate. The spatially adaptive Gaussian kernel employed, denoted $W_i(x, y)$, is defined as

$$W_i(x, y) = \mathbf{K} e^{-\frac{1}{2}(\mathbf{y}-\mathbf{x})^T \mathbf{U}_{i-1}^T \Lambda_i^{-2} \mathbf{U}_{i-1}(\mathbf{y}-\mathbf{x})} \quad (3.10)$$

where

$$\Lambda_i = \begin{bmatrix} \sqrt{2}R_i^1 & 0 \\ 0 & \sqrt{2}R_i^2 \end{bmatrix} \quad (3.11)$$

and K is a scalar associated with the Gaussian function. Initially, the Gaussian window is isotropic to begin computation. U_i contains the eigenvectors (e_i^1, e_i^2) , U_{-1} is initialized to the coordinate axes, and $\lambda_i^1 > \lambda_i^2$ are the eigenvalues of the structure tensor at the i^{th} iteration. The Gaussian window size is updated based on the eigenvalues at each iteration as

$$R_{i+1}^1 = \frac{\lambda_i^1}{\lambda_i^1 + \lambda_i^2} R_i^1 \quad (3.12)$$

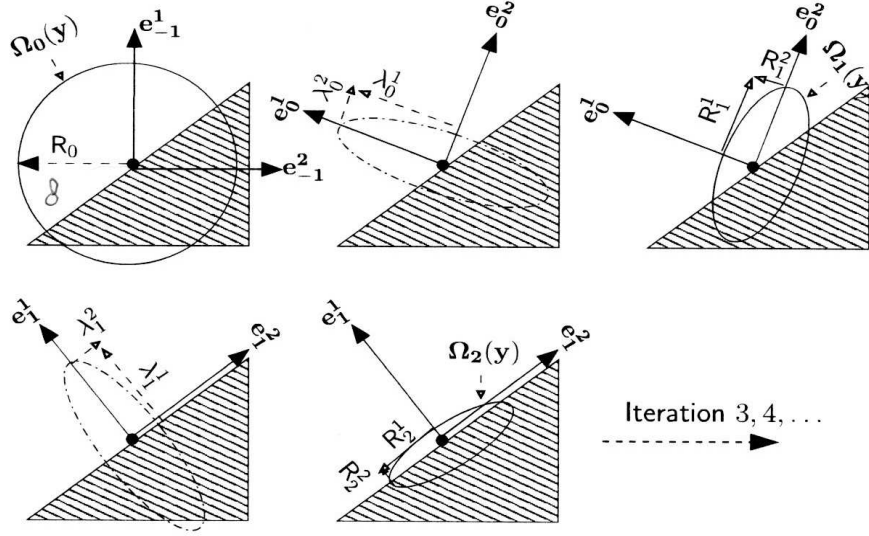


Fig. 3.2: The first iterations of the spatially adaptive Gaussian window algorithm used in the adaptive structure tensor computation. R_i^1, R_i^2 are the semi-major and minor axes, respectively, at iteration i and Ω_i is the corresponding neighborhood.

and

$$R_{i+1}^2 = \frac{\lambda_i^2}{\lambda_i^1 + \lambda_i^2} R_i^2 \quad (3.13)$$

The kernel is also updated by the following equation

$$W_{i+1}(x, y) = K e^{-\left(\frac{1}{2}(y-x)^T U_i^T \Lambda_{i+1}^{-2} U_i (y-x)\right)} \quad (3.14)$$

and

$$\Lambda_{i+1} = \begin{bmatrix} \sqrt{2}R_{i+1}^1 & 0 \\ 0 & \sqrt{2}R_{i+1}^2 \end{bmatrix} \quad (3.15)$$

The kernel equation is now modified to account for the adaptive kernel as

$$J(x, v_i, W_{i+1})v_{i+1} = \lambda v_{i+1} \quad (3.16)$$

The above presented technique is employed to adapt the extensor window size

and orientation in the ABE interpolation algorithm. The iterative process depicting the adaptive spatial kernel size is shown in Figure 3.2. Details of the algorithm and incorporation of the adaptive structure tensor are presented in the following chapter.

Chapter 4

Adaptive Bilateral Extensor (ABE)

Interpolation

Overview

Bilateral filtering adapts to local tonal structure and will be incorporated in the proposed adaptive bilateral extensor algorithm. Possible bilateral kernel functions are next described. It is followed by a novel image interpolation algorithm based on the adaptive bilateral filter with results and discussion. An adaptive bilateral formulation of extensor-based interpolation is proposed that is able to dynamically adapt the extensor kernel based on the image structure and image intensity information within an adaptive neighborhood of the current pixel location.

4.1 Bilateral Filtering

THE *bilateral filter* was originally proposed in [21] and is outlined as follows. Bilateral filtering was principally designed to remove noise from an image. Simple methods for denoising, such as low-pass filtering, blur edges in the resulting image [21]. Depending on the amount of filtering required, significant distortion of

the original image may be present in the processed result. More advanced methods including anisotropic diffusion [22] require an iterative process and significant computational cost to denoise an image. Bilateral filtering, as proposed in [21], is formulated as a noniterative process that only requires a single pass over an image.

The bilateral filter removes background noise while preserving the structural aspects of an image by considering both the spatial and intensity domains. Following the original notation of [21], a low-pass domain filter is formulated as

$$\mathbf{h}(x) = k_d^{-1}(x) \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} \mathbf{f}(\xi) c(\xi, x) d\xi \quad (4.1)$$

where

$$k_d(x) = \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} c(\xi, x) d\xi \quad (4.2)$$

is a normalization term and $c(\xi, x)$ is a kernel function that measures the spatial closeness between the center position x within a region and ξ that resides at a nearby point. The second component consists of a range filter defined by

$$\mathbf{h}(x) = k_r^{-1}(x) \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} \mathbf{f}(\xi) s(\mathbf{f}(\xi), \mathbf{f}(x)) d\xi \quad (4.3)$$

where

$$k_r(x) = \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} s(\mathbf{f}(\xi), \mathbf{f}(x)) d\xi \quad (4.4)$$

corresponds to the normalization term and $s(\mathbf{f}(\xi), \mathbf{f}(x))$ represents a kernel function that measures the closeness of pixel intensity between the center pixel of a neighborhood, $\mathbf{f}(x)$, and a nearby pixel $\mathbf{f}(\xi)$ within the region. Each of the above filters contains desirable properties for noise removal. By combining them together, we filter the image based on the spatial distance and intensity difference.

The domain filter will intuitively show preference to those pixels located nearby

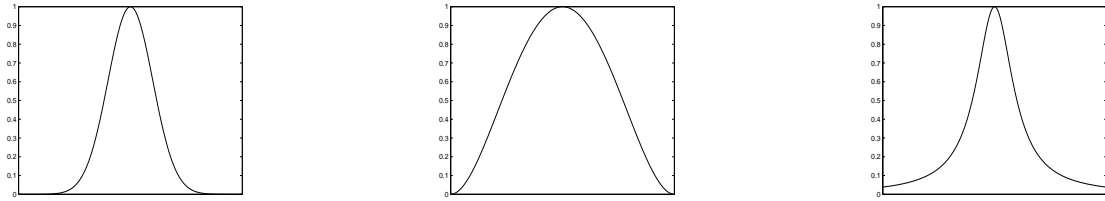


Fig. 4.1: Comparison of (left to right) Gaussian, Tukey, and Geman-McClure kernel functions

in terms of spatial distance. However, the range filter will measure closeness of pixels in the region based on the difference in pixel intensity. If the region contains an edge boundary, only those pixels whose intensity is close to that of the center pixel will be given preference, while the pixels on the opposite side of the edge boundary will be biased against. Combining together the properties of the domain and range filters yields the bilateral filter

$$h(x) = k^{-1}(x) \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} f(\xi) c(\xi, x) s(f(\xi), f(x)) d\xi \quad (4.5)$$

and normalization factor k defined as

$$k(x) = \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} c(\xi, x) s(f(\xi), f(x)) d\xi \quad (4.6)$$

Several possibilities exist to represent the kernel weighting functions employed in the bilateral filter. The original formulation [21] employs a Gaussian kernel computed as

$$w(x, \sigma) = e^{-\frac{1}{2} \left(\frac{x}{\sigma}\right)^2} \quad (4.7)$$

Some other well-known candidates are the Tukey robust estimator [20] defined as



Fig. 4.2: 512x512 Lena Sjöblom with areas utilized for bilateral filtering.

$$w(x, \sigma) = \left\{ \begin{array}{ll} \left(1 - \left(\frac{x}{\sigma}\right)^2\right)^2 & \text{if } |x| \leq \sigma \\ 0 & \text{otherwise} \end{array} \right\} \quad (4.8)$$

and the Geman-McClure function [20] formulated as

$$w(x, \sigma) = 1 - \frac{x^2}{\sigma^2 + x^2} \quad (4.9)$$

In each of the above kernel formulations, σ is the standard deviation and x is the input value to the filter (pixel intensities or spatial distance) for the bilateral filter. Figure 4.1 graphically illustrates each of the candidate kernels. In combining the spatial and intensity domain filters, σ_d denotes the value employed for filtering in the spatial domain and σ_i denotes usage in the intensity domain. To preserve the color table of the original image, σ_i values combine all colors channels together for filtering. A textured region of Lena's hat in Figure 4.2 is utilized to compare the kernel function's performance with $\sigma_d = 10$ and several values of σ_i as shown in Figure 4.3. The Tukey kernel function is chosen for use in this thesis for its ability

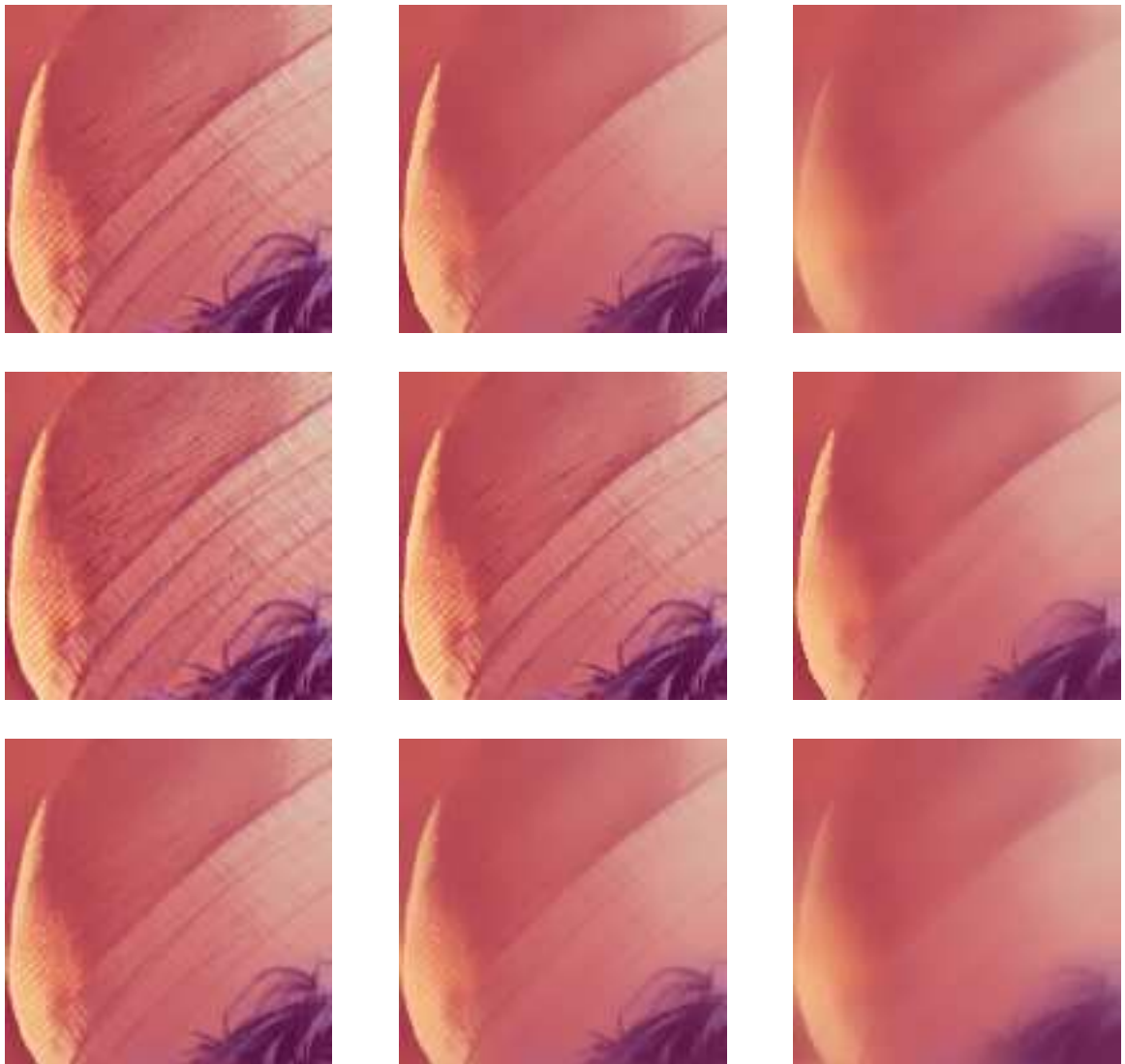


Fig. 4.3: Comparison of (top to bottom) Gaussian, Tukey, and Geman McClure kernels for bilateral filtering with $\sigma_d = 10$ and $\sigma_i = 10, 30, 100$ in each of the columns (left to right) respectively.

to retain features and textures better than that of the Gaussian or Geman-McClure kernels as evidenced in the above comparison.

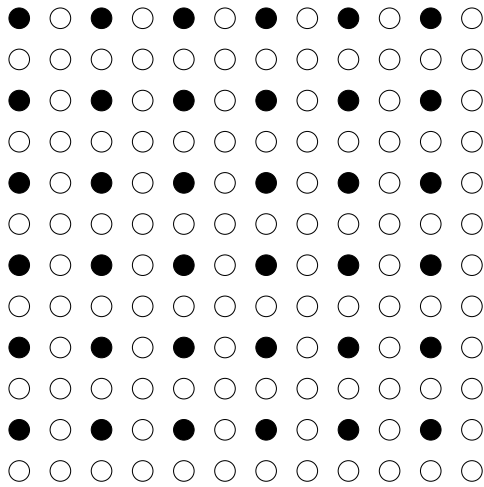


Fig. 4.4: Depiction of original pixels, denoted in black, in the interpolated result.

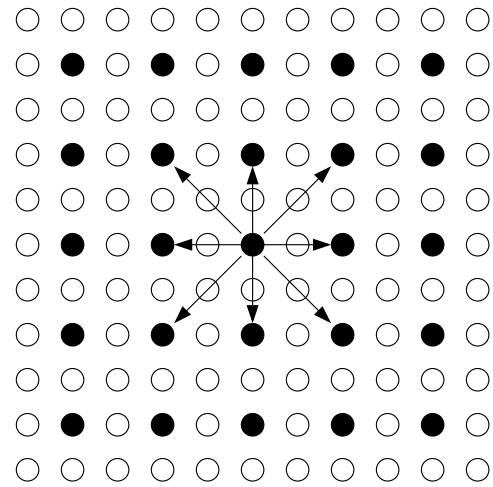


Fig. 4.5: Computation of bilateral weights for I_{xy} .

4.2 Bilateral Interpolation Algorithm

One major drawback of current interpolation methods is the aliasing incurred along edge boundaries. It follows that the bilateral filter has the potential to produce improved results as compared to current interpolation methods due to its edge preserving properties. The following is a adaptation of the bilateral filter to image interpolation hereafter known as *bilateral interpolation*.

Interpolation begins by distributing the original pixels from the input image evenly as illustrated in Figure 4.4 where black represents the original pixels and white denotes interpolated pixels. Next, the bilateral filter weights are computed using the Tukey kernel on the original image. The weights utilized for each pixel within the interpolation window, usually 3x3, are computed and stored such that each original pixel value is associated with its bilateral filter weights as shown in Figure 4.5. Next, interpolation proceeds for the unknown pixels in the supersampled space. An unknown pixel intensity I'_{xy} is interpolated using the bilateral filter weights from the nearest original pixel. The interpolated value is obtained from convolving the original



Fig. 4.6: A 128x128 portion of Lena processed using bilateral interpolation (without extensor component) by a factor of two with $\sigma_i = 90$.

pixels in the interpolation window with the bilateral weights.

4.3 Bilateral Interpolation Results

Results of bilateral interpolation are illustrated in the following. Interpolation is first performed using a center 128x128 region of the decimated Lena image in order to compare the result with current interpolation methods. As shown in Figure 4.6, the aliasing present in the decimated input image is enhanced, especially along the brim of the hat and the shoulder. This is clearly not desirable and is of poor visual quality. However, advantage can be taken of this since the aliased edge boundaries present in the input image are well preserved. While not suitable for low quality images, employing a high quality input image, free of aliasing, may produce a visually superior result. Figure 4.7 illustrates bilateral interpolation using the original Lena image. The result is a high quality interpolated image equivalent to that of bicubic

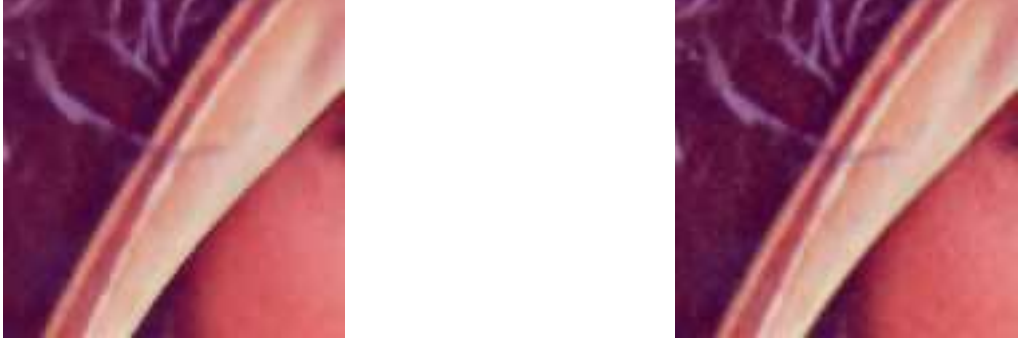


Fig. 4.7: Hat brim region of the original Lena image interpolated by a factor of 2 using bilateral interpolation (left) with $\sigma_i = 120$, $\sigma_d = 10$ and bicubic interpolation (right)

interpolation. However, a method which can interpolate a decimated image without producing aliasing artifacts is needed. The edge-preserving property of the bilateral filter inspires development of a new extensor-based interpolation algorithm formulated in the following section.

4.4 Adaptive Bilateral Extensor Interpolation Algorithm

A novel algorithm for image interpolation known as Adaptive Bilateral Extensor (ABE) interpolation is presented in this thesis. The bilateral filter, $B(\vec{x})$, utilized for extensor-based image interpolation is defined as

$$B(\vec{x}) = \frac{1}{k(\vec{x})} \sum_{\vec{p} \in \Omega} W(\vec{x} - \vec{p}, \sigma_d) g(I_{\vec{p}} - I_{\vec{x}}, \sigma_i) I_{\vec{p}} \quad (4.10)$$

where $W()$ is the spatial kernel obtained from the adaptive structure tensor and $g()$ is the tonal weighting function based on intensity difference and $I_{\vec{p}}$ is the image intensity at location \vec{p} . The Tukey robust kernel function is used for $W()$ and $g()$. The tuning parameters σ_i , σ_d weight the respective distances in the kernel functions such that smaller values give less weight to outliers. The bilateral filter is incorporated into the

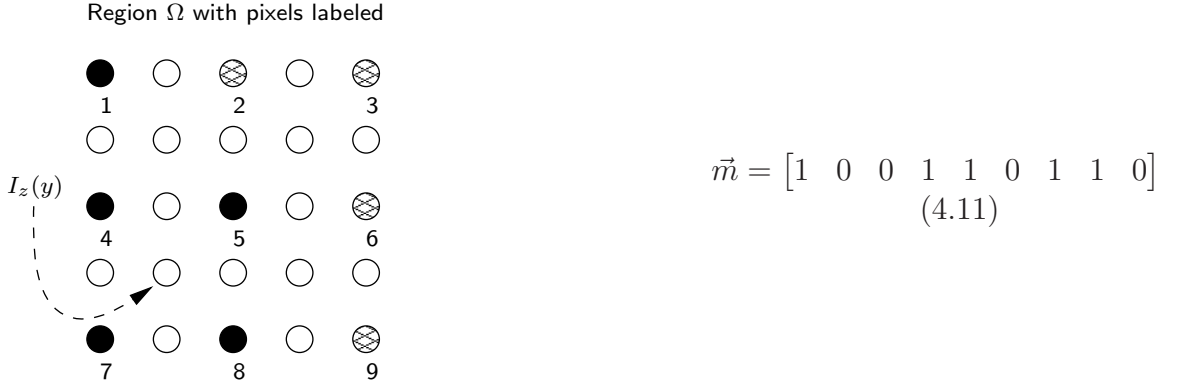


Fig. 4.8: Mask \vec{m} based on included pixels (in black), excluded pixels (patterned), pixels to interpolate (in white). The pixel to be interpolated is identified as $I_z(y)$ and is within the group of black labeled pixels in this example.

extensor formulation by modifying Eq 2.9 as

$$\begin{aligned} I_z(\vec{y}) &= \mathbf{T}\varphi_y \\ &= \left(\tilde{\mathbf{G}}\mathbf{I}_{\vec{m}} \right)_{\downarrow} \left(\vec{m}\vec{m}^T\mathbf{X} \right)_{\downarrow\rightarrow}^{-1} \left(\mathbf{I}_{\vec{m}}\vec{\varphi}_y \right)_{\rightarrow} + \vec{\mu} \end{aligned} \quad (4.12)$$

$$= \hat{\mathbf{G}}\mathbf{X}^{-1}\hat{\varphi}_y + \mu \quad (4.13)$$

In Eq 4.13, $\downarrow\rightarrow$ subscript represents the operator that removes zero columns and zero rows from a matrix, \downarrow is the operator that removes corresponding zero columns, and \rightarrow represents the operator that removes corresponding zero rows from a vector. The vector \vec{m} represents a mask computed within region Ω in lexicographical order based on the bilateral filter $B(x)$ that identifies which pixels will be used within the extensor interpolation stage.

An example is shown in Figure 4.8 for a 3x3 region Ω in which there is an edge structure. $I_{\vec{m}}$ is a diagonal matrix where the diagonal elements are from mask vector \vec{m} in the image, so that the pixels can be grouped into two regions (labeled black and

gray). The mask vector \vec{m} is computed as shown below:

$$m_i = \begin{cases} 1 & \text{if } B(\vec{x}) \leq \sigma_i \\ 0 & \text{otherwise} \end{cases} \quad (4.14)$$

where

$$\sigma_i = \frac{1}{2b} \sum_{i=1}^b \left(\max I_{\vec{p}}^{(i)} - \min I_{\vec{p}}^{(i)} \right) \quad (4.15)$$

with the additional condition that $\vec{p} \in \Omega$ for $\max I_{\vec{p}}^{(i)}$ and $\min I_{\vec{p}}^{(i)}$. The bilateral filter value $B(\vec{x})$ is used as a threshold value for each pixel location \vec{x} (or lexicographical position i in mask \vec{m}_i) to determine if the pixel is to be included or excluded from the extensor. The tonal tuning parameter σ_i is automatically computed using half the average range of pixel intensities for each channel within region Ω as shown in Eq 4.15. We can write Eq 4.12 in a simplified form as

$$I_z(\vec{y}) = \left(\tilde{\mathbf{G}}\mathbf{I}_{\vec{m}} \right)_{\downarrow} \left(\vec{m}\vec{m}^T\mathbf{X} \right)_{\downarrow}^{-1} \left(\mathbf{I}_{\vec{m}}\vec{\varphi}_y \right)_{\rightarrow} + \vec{\mu} \quad (4.16)$$

$$= \hat{\mathbf{G}}\hat{\mathbf{X}}^{-1}\hat{\varphi}_y + \mu \quad (4.17)$$

where $\hat{\mathbf{G}} = \mathbf{I}_m$, $\hat{\mathbf{X}} = \vec{m}\vec{m}^T\mathbf{X}$, $\hat{\varphi}_y = I_m\varphi_y$. The extensor matrix \mathbf{X} for this example is shown in Eq 4.18 corresponding to the region Ω from Figure 4.8.

$$\mathbf{X} = \begin{bmatrix} \varphi_{11} & \varphi_{12} & \varphi_{13} & \varphi_{14} & \varphi_{15} & \varphi_{16} & \varphi_{17} & \varphi_{18} & \varphi_{19} \\ \varphi_{21} & \varphi_{22} & \varphi_{23} & \varphi_{24} & \varphi_{25} & \varphi_{26} & \varphi_{27} & \varphi_{28} & \varphi_{29} \\ \varphi_{31} & \varphi_{32} & \varphi_{33} & \varphi_{34} & \varphi_{35} & \varphi_{36} & \varphi_{37} & \varphi_{38} & \varphi_{39} \\ \varphi_{41} & \varphi_{42} & \varphi_{43} & \varphi_{44} & \varphi_{45} & \varphi_{46} & \varphi_{47} & \varphi_{48} & \varphi_{49} \\ \varphi_{51} & \varphi_{52} & \varphi_{53} & \varphi_{54} & \varphi_{55} & \varphi_{56} & \varphi_{57} & \varphi_{58} & \varphi_{59} \\ \varphi_{61} & \varphi_{62} & \varphi_{63} & \varphi_{64} & \varphi_{65} & \varphi_{66} & \varphi_{67} & \varphi_{68} & \varphi_{69} \\ \varphi_{71} & \varphi_{72} & \varphi_{73} & \varphi_{74} & \varphi_{75} & \varphi_{76} & \varphi_{77} & \varphi_{78} & \varphi_{79} \\ \varphi_{81} & \varphi_{82} & \varphi_{83} & \varphi_{84} & \varphi_{85} & \varphi_{86} & \varphi_{87} & \varphi_{88} & \varphi_{89} \\ \varphi_{91} & \varphi_{92} & \varphi_{93} & \varphi_{94} & \varphi_{95} & \varphi_{96} & \varphi_{97} & \varphi_{98} & \varphi_{99} \end{bmatrix}_{9 \times 9} \quad (4.18)$$

The actual mask for pixel $I_z[i]$ from Figure 4.8 is shown in Eq 4.19. To illustrate

the $\downarrow \rightarrow$ operator, Eq 4.20 shows the result of removing the zero rows and columns.

$$(\vec{m}\vec{m}^T\mathbf{X}) = \begin{bmatrix} \varphi_{11} & 0 & 0 & \varphi_{14} & \varphi_{15} & 0 & \varphi_{17} & \varphi_{18} & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ \varphi_{41} & 0 & 0 & \varphi_{44} & \varphi_{45} & 0 & \varphi_{47} & \varphi_{48} & 0 \\ \varphi_{51} & 0 & 0 & \varphi_{54} & \varphi_{55} & 0 & \varphi_{57} & \varphi_{58} & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ \varphi_{71} & 0 & 0 & \varphi_{74} & \varphi_{75} & 0 & \varphi_{77} & \varphi_{78} & 0 \\ \varphi_{81} & 0 & 0 & \varphi_{84} & \varphi_{85} & 0 & \varphi_{87} & \varphi_{88} & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}_{9 \times 9} \quad (4.19)$$

$$\hat{\mathbf{X}} = (\vec{m}\vec{m}^T\mathbf{X})_{\downarrow \rightarrow} = \begin{bmatrix} \varphi_{11} & \varphi_{14} & \varphi_{15} & \varphi_{17} & \varphi_{18} \\ \varphi_{41} & \varphi_{44} & \varphi_{45} & \varphi_{47} & \varphi_{48} \\ \varphi_{51} & \varphi_{54} & \varphi_{55} & \varphi_{57} & \varphi_{58} \\ \varphi_{71} & \varphi_{74} & \varphi_{75} & \varphi_{77} & \varphi_{78} \\ \varphi_{81} & \varphi_{84} & \varphi_{85} & \varphi_{87} & \varphi_{88} \end{bmatrix}_{5 \times 5} \quad (4.20)$$

The pixel intensity represented by $\tilde{\mathbf{G}}$ and subsequently multiplied by \mathbf{I}_m , denoted as $\hat{\mathbf{G}}$, is computed similar to that of $\hat{\mathbf{X}}$ as

$$\hat{\mathbf{G}} = \tilde{\mathbf{G}}\mathbf{I}_m = \begin{bmatrix} G_{r,1} - \mu_r & G_{r,2} - \mu_r & \cdots & G_{r,9} - \mu_r \\ G_{g,1} - \mu_g & G_{g,2} - \mu_g & \cdots & G_{g,9} - \mu_g \\ G_{b,1} - \mu_b & G_{b,2} - \mu_b & \cdots & G_{b,9} - \mu_b \end{bmatrix}_{3 \times 9} \begin{bmatrix} 1 & 0 & 0 & 1 & 1 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 1 & 1 & 0 & 1 & 1 & 0 \\ 1 & 0 & 0 & 1 & 1 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 1 & 1 & 0 & 1 & 1 & 0 \\ 1 & 0 & 0 & 1 & 1 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}_{9 \times 9} \quad (4.21)$$

$$\hat{\mathbf{G}} = \begin{bmatrix} G_{r,1} - \mu_r & 0 & 0 & G_{r,4} - \mu_r & G_{r,5} - \mu_r & 0 & 0 & G_{r,7} - \mu_r & G_{r,8} - \mu_r & 0 \\ G_{g,1} - \mu_g & 0 & 0 & G_{g,4} - \mu_g & G_{g,5} - \mu_g & 0 & 0 & G_{g,7} - \mu_g & G_{g,8} - \mu_g & 0 \\ G_{b,1} - \mu_b & 0 & 0 & G_{b,4} - \mu_b & G_{b,5} - \mu_b & 0 & 0 & G_{b,7} - \mu_b & G_{b,8} - \mu_b & 0 \end{bmatrix}_{3 \times 9} \quad (4.22)$$

$$\hat{\mathbf{G}}_{\downarrow} = \begin{bmatrix} G_{r,1} - \mu_r & G_{r,4} - \mu_r & G_{r,5} - \mu_r & G_{r,7} - \mu_r & G_{r,8} - \mu_r \\ G_{g,1} - \mu_g & G_{g,4} - \mu_g & G_{g,5} - \mu_g & G_{g,7} - \mu_g & G_{g,8} - \mu_g \\ G_{b,1} - \mu_b & G_{b,4} - \mu_b & G_{b,5} - \mu_b & G_{b,7} - \mu_b & G_{b,8} - \mu_b \end{bmatrix}_{3 \times 5} \quad (4.23)$$

Equation 4.12 for solving for an interpolated pixel value can also be expressed

using permutation matrices. For example,

$$\begin{bmatrix} a & b & c \\ d & e & f \\ g & h & i \end{bmatrix} \begin{bmatrix} 0 & 1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} b & a & c \\ e & d & f \\ h & g & i \end{bmatrix} \quad (4.24)$$

The permutation matrices allow the nonzero elements to move to the top-left corner of the matrix and the zero elements to shift to the bottom. The equation to solve for an interpolated pixel value can be expressed using permutation matrices as

$$I_z(\vec{y}) = \left(\tilde{\mathbf{G}} I_m \right)_{\downarrow} (m m^T \mathbf{X}^{-1})_{\downarrow \rightarrow} (\mathbf{I}_{\vec{m}} \vec{\varphi}_y)_{\rightarrow} + \vec{\mu} \quad (4.25)$$

$$= \hat{\mathbf{G}} \hat{\mathbf{X}}^{-1} \hat{\varphi}_y + \mu \quad (4.26)$$

$$= \left(\hat{\mathbf{G}} P^T \right)_{\downarrow} (P \mathbf{X} P)^{-1}_{\downarrow \rightarrow} (P \varphi_y)_{\rightarrow} + \mu \quad (4.27)$$

where

$$P = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}_{9 \times 9} \quad (4.28)$$

Note that $P^{-1} = P^T$ using the above notation. In Eq 4.16, if l is the number of zero elements in vector \vec{m} , corresponding to the last l columns (indicating the number of excluded pixels from the extensor) then \downarrow indicates to remove the last l columns and \rightarrow indicates removal of the last l rows. The permutation matrix in Eq 4.28 shifts the non-zero values to the left and top of the matrix. The remaining zero rows and columns are subsequently removed by the $\downarrow \rightarrow$ operator.

Detection and orientation of edges throughout the image is calculated as explained in Chapter 3. The strength of an edge boundary is calculated from the eigenvalues as

$$\tau_{edge} = \sqrt{\lambda_1 - \lambda_2} \quad (4.29)$$

where τ_{edge} is the confidence signifying the strength of the gradient. The orientation angle θ is obtained from v_1 and v_2 by

$$\theta = -\arctan\left(\frac{v_2}{v_1}\right) - \frac{\pi}{2}, \quad \theta \in [0, \pi] \quad (4.30)$$

which produces the orientation of the gradient at a given pixel. The $\frac{\pi}{2}$ term is employed to obtain the orientation along the edge. θ is then used to orient the extensor window, enabling more information from near the edge to be incorporated. Interpolation of a given pixel proceeds as follows. If τ_{edge} is less than a predetermined threshold for a given pixel, it not a sufficiently strong edge and we employ a standard extensor window of 5x5 centered at x_{ij} to interpolate its color intensity. In the case that τ_{edge} meets or exceeds the threshold, an elliptical extensor region β is employed as follows. The size of the extensor region with major radius α_1 and minor radius α_2 is calculated as

$$\alpha_1 = \frac{1}{\sqrt{\lambda_1}}, \quad \alpha_2 = \frac{1}{\sqrt{\lambda_2}} \quad (4.31)$$

where the size is computed with respect to the interpolated domain. It is then rotated by angle θ to orient the extensor along the edge as shown in Figure 4.9 where filled dots represent original pixels and white dots represent values to interpolate. The black and patterned dots illustrate an edge boundary.

Interpolation proceeds via a multi-step process. It begins by interpolating the specified image, I , using an isotropic extensor to obtain an initial estimate of the in-

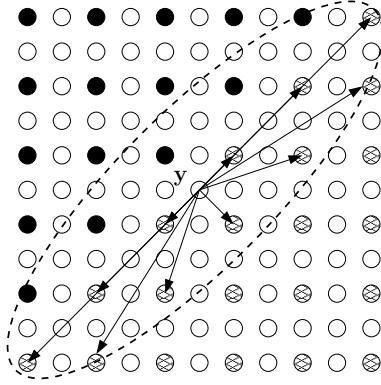


Fig. 4.9: Pixels included for interpolation using adaptive bilateral extensor

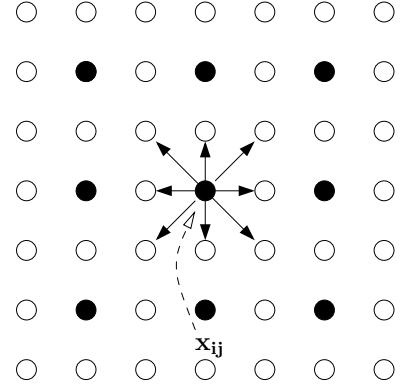


Fig. 4.10: Pixels included in regularization of x_{ij} .

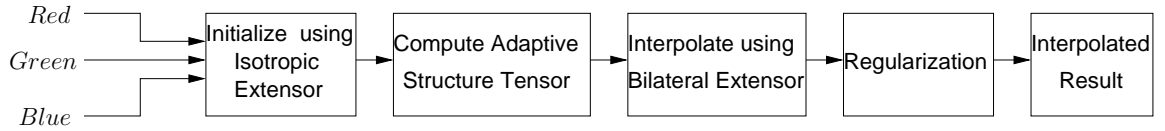


Fig. 4.11: Diagram of Adaptive Bilateral Extensor Interpolation algorithm.

terpolated values in the supersampled domain. This estimate is then used to initialize the bilateral filter. Computation continues until

$$\delta^{(k+1)}I = |I^{(k+1)} - I^{(k)}| \leq T \quad (4.32)$$

Figures 4.12 and 4.13 illustrate the convergence of the actual edge direction as the process iterates. Once the intensity values have converged for each interpolated pixel, the process is transposed and new intensity values are computed for each of the original pixel values in the image using the interpolated intensities computed previously. This final step ensures that original pixel intensities are smoothly integrated with the interpolated pixel values as depicted in Figure 4.10. A diagram of the interpolation process is shown in Figure 4.11. The complete interpolation algorithm is outlined as follows:

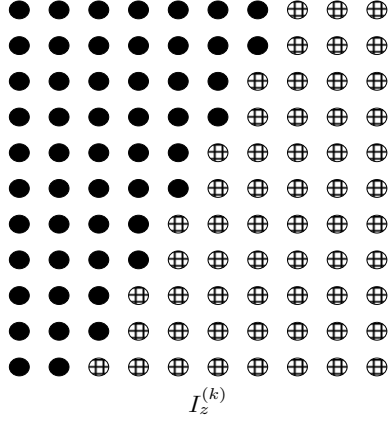


Fig. 4.12: Estimate of edge at iteration I^k

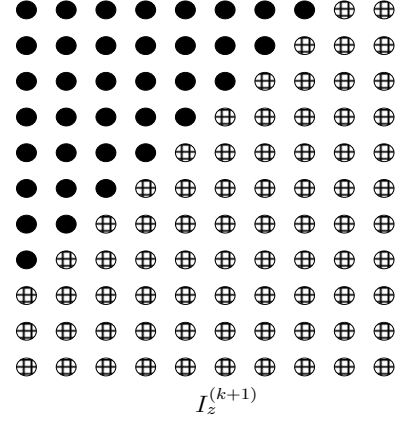


Fig. 4.13: Estimate of edge at iteration $I^{(k+1)}$

4.4.1 Adaptive Bilateral Extensor Algorithm

```

1: //First pass through image to compute adaptive structure tensor
2: for each original pixel in the image do
3:     Compute adaptive structure tensor [see Chapter 3] to identify local image
       structure parameters  $[\lambda_1, \lambda_2]$ 
4:     //Estimate local edge strength
5:      $\tau_{edge} \leftarrow \sqrt{\lambda_1 - \lambda_2}$ 
6: end for
7: //Second pass through image to selectively apply ABE interpolation
8: for each interpolated pixel do
9:     i  $\leftarrow$  Position of next pixel in  $\Omega$ 
10:    if  $\tau_{edge} \geq 0.001$  then
11:        //Use ABE interpolation near strong image gradients
12:        extensorWidth  $\leftarrow \frac{1}{\sqrt{\lambda_2[i]}}$ 
13:        extensorHeight  $\leftarrow \frac{1}{\sqrt{\lambda_1[i]}}$ 
14:        //Compute  $\vec{m}$  within the extensor window
15:        for each original pixel i in the extensor window do

```

```

16:          $\sigma_i = \frac{1}{2b} \sum_{i=1}^b \left( \max I_{\vec{p}}^{(i)} - \min I_{\vec{p}}^{(i)} \right)$ 
17:          $\vec{m}_{[i]} \leftarrow \mathbf{B}(\vec{x}) = \frac{1}{k(\vec{x})} \sum_{\vec{p} \in \Omega} \mathbf{W}(\vec{x} - \vec{p}, \tau_{edge}) \mathbf{g}(\mathbf{I}_{\vec{p}} - \mathbf{I}_{\vec{x}}, \sigma_i) I_{\vec{p}}$ 
18:     end for
19:      $I_z[i] \leftarrow \mathbf{T}\varphi[i] = \left( \mathbf{I}_{\vec{m}} \tilde{\mathbf{G}} \right)_{\downarrow} \left( \vec{m} \vec{m}^T \mathbf{X} \right)_{\downarrow \rightarrow}^{-1} \left( \mathbf{I}_{\vec{m}} \cdot \varphi[i] \right)_{\rightarrow} + \vec{\mu}$ 
20: else
21:     //Use isotropic window size
22:     extensorWidth  $\leftarrow 5$ 
23:     extensorHeight  $\leftarrow 5$ 
24:      $I_z[i] \leftarrow \mathbf{T}\varphi[i] = \tilde{\mathbf{G}} \mathbf{X}^{-1} \varphi[i] + \vec{\mu}$ 
25: end if
26: end for
27: //Perform regularization on original pixels
28: extensorWidth  $\leftarrow 3$ 
29: extensorHeight  $\leftarrow 3$ 
30: while all original pixels not regularized do
31:     if  $I_i$  is original then
32:          $I_z[i] \leftarrow \mathbf{T}\varphi[i] = \tilde{\mathbf{G}} \mathbf{X}^{-1} \varphi[i] + \vec{\mu}$ 
33:     end if
34: end while

```

Chapter 5

Performance Evaluation Metrics and Results

Overview

Results of the ABE interpolation algorithm are presented in this chapter. It begins with a discussion of the desirable characteristics sought in an evaluation technique. Next, evaluation metrics utilized in this thesis are introduced. Discussion of the ABE interpolation algorithm and a critical comparison to other state-of-the-art techniques are then presented. The chapter concludes by examining the effect of regularization on the ABE algorithm.

5.1 Image Evaluation Metrics

IN assessing the quality of an interpolated image, objective metrics are desired to combine with a subjective assessment in order to obtain a suite of quality metrics for evaluation. Methods including mean squared error (MSE) and signal to noise ratio (SNR) are convenient to compute. While they produce a tangible result, they do not accurately capture perceptual or structural quality. It is desirable to

have an objective method independent of human assessment that is sensitive these qualities.

Objective metrics capturing elements of the human visual system are currently an active area of research. Current methods of quality assessment discuss the need to develop a metric which models the human visual system's (HVS) own assessment of image quality. One recent method formulates a new technique combining loss of correlation with mean and variance distortion to capture structural differences, rather than attempting to base quality on a combination of pixel differences [23].

A technique known as structural similarity (SSIM) achieves an objective metric for evaluation by separating the measurement into luminance, contrast, and structure [24]. The three components are then assessed by a comparison function and combined to produce the resulting similarity index value.

Three facets of comparison are employed in this thesis. Holding with traditional techniques, SNR [25] is computed as

$$SNR = 10 \log_{10} \frac{\sum_{i=0}^M \sum_{j=0}^N I(i, j)^2}{\sum_{i=0}^M \sum_{j=0}^N (I(i, j) - I'(i, j))^2} \quad (5.1)$$

where $I(i, j)$ is the pixel from the original image, $I'(i, j)$ is the pixel value from the interpolated image, and M, N are the height and width, respectively of the images. Multi-channel images are first converted to grayscale before comparison using

$$I_{grayscale} = 0.299I_{red} + 0.587I_{green} + 0.114I_{blue} \quad (5.2)$$

which is a standard RGB to grayscale conversion as defined by the National Television System Committee (NTSC) [26] and $I_{channel}$ represents the pixel value of a specific color channel. This provides a familiar result consistent with many existing works. Also employed is a method of comparison consisting of a human visual assess-

ment, providing feedback and analysis beyond a quantitative assessment of quality. The above described SSIM metric is also included for structural quality evaluation independent of a human visual inspection and is described in the following section.

5.1.1 Structural Similarity Quality Evaluation

The SSIM image quality evaluation metric provides a numerical comparison between an original and distorted image signal, resulting in a measure of the perceived quality of the distorted image with reference to the original, expressed as a percentage of closeness. Following the notation of [24], a perfect quality image x is compared to a distorted signal y using components of luminance, contrast, and structure. The luminance measure is defined as

$$l(x, y) = \frac{2\mu_x\mu_y + C_1}{\mu_x^2 + \mu_y^2 + C_1} \quad (5.3)$$

where

$$\mu = \frac{1}{N} \sum_{i=1}^N x_i \quad (5.4)$$

and μ represents the mean intensity of the image. $C_1 = (K_1L)^2$ is a small constant value to maintain numerical stability and L is the number of colors in the image. The HVS is more sensitive to relative luminance change as compared to absolute luminance, which is shown above. Therefore the luminance measure is altered to account for this by allowing $\mu_y = (1 + R)\mu_x$ leading to

$$l(x, y) = \frac{2(1 + R)}{1 + (1 + R)^2 + \frac{C_1}{\mu_x^2}} \quad (5.5)$$

where R represents the size of luminance change relative to background luminance. The contrast comparison function is defined similar to that of luminance, except that

the standard deviation of mean intensity is employed as a measure of contrast by

$$\sigma_x = \left(\frac{1}{N-1} \sum_{i=1}^N (x_i - \mu_x)^2 \right)^{\frac{1}{2}} \quad (5.6)$$

The contrast comparison function is then defined as

$$c(x, y) = \frac{2\sigma_x\sigma_y + C_2}{\sigma_x^2 + \sigma_y^2 + C_2} \quad (5.7)$$

where C_2 is defined similar to C_1 as discussed above. The structural component is evaluated after variance normalization and luminance subtraction. The structural correlation is specifically defined as the dot product between the normalized, luminance subtracted signals as

$$s(x, y) = \frac{\sigma_{xy} + C_3}{\sigma_x\sigma_y + C_3} \quad (5.8)$$

where

$$\sigma_{xy} = \frac{1}{N-1} \sum_{i=1}^N (x_i - \mu_x)(y_i - \mu_y) \quad (5.9)$$

and commonly $C_3 = \frac{C_2}{2}$. The overall SSIM metric can now be defined as

$$SSIM(x, y) = [l(x, y)]^\alpha [c(x, y)]^\beta [s(x, y)]^\gamma \quad (5.10)$$

where $[\alpha, \beta, \gamma] > 0$ are weights that allow emphasis to be placed on different components. A common weight distribution is uniform with $[\alpha, \beta, \gamma] = 1$. Additionally, $K_1 = 0.01$, $K_2 = 0.03$, and $K_3 = 0.015$ for the implementation utilized here. Previous works on structural similarity [27, 28] use a fixed size square window by which the metric is computed. It is observed that this produces "blockiness" in the index map near the window edges. To prevent this, an 11x11 Gaussian weighting function ($\sum_{i=1}^N w_i = 1$) is utilized, requiring the following modification to μ_x , σ_x , and σ_{xy}

respectively

$$\mu_x = \sum_{i=1}^N w_i x_i \quad (5.11)$$

$$\sigma_x = \left(\sum_{i=1}^N w_i (x_i - \mu_x)^2 \right)^{\frac{1}{2}} \quad (5.12)$$

$$\sigma_{xy} = \sum_{i=1}^N w_i (x_i - \mu_x) (y_i - \mu_y) \quad (5.13)$$

The mean value of the overall SSIM index map is taken to provide a single value for an overall measure of quality between image signals X and Y denoted as

$$Q(X, Y) = \frac{1}{M} \sum_{j=1}^M SSIM(x_j, y_j) \quad (5.14)$$

5.2 Interpolation Results

The ABE interpolation algorithm is qualitatively compared to bicubic [13], edge-directed [29], isotropic extensor (IE) [12], and the original images from which a decimated source image is obtained. Numerical metrics SNR and SSIM [30] are also employed for a numerical comparison. The house, airplane (F-16), and red brick house images are utilized for detailed analysis, shown in Figures 5.2, 5.21, and 5.29, respectively with regions further utilized as labeled. All images are obtained from the USC Signal and Image Processing Institute's image repository [31]. The original images are first subsampled by a factor of two in each direction by removing the even rows and columns as shown in Figure 5.1. The decimated images are then interpolated using the above described methods. A size of 7x7 is employed for computation involving the IE algorithm. The implementation of bicubic interpolation is performed using Matlab. Table 5.1 summarizes the parameters employed for interpolation using the IE and ABE algorithms. Other methods utilized for comparison use default

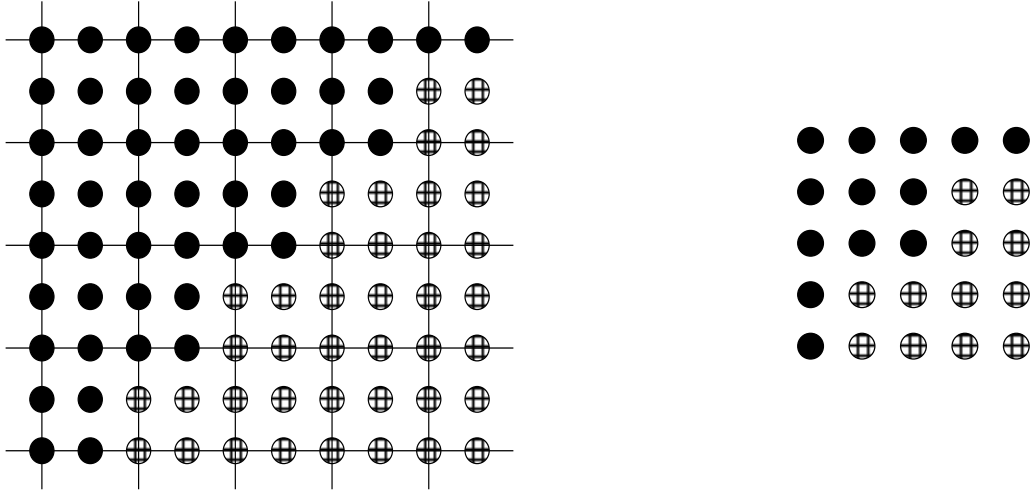


Fig. 5.1: Depiction of decimation employed before interpolation. The original image with rows and columns marked for removal (left). Resulting decimated image (right).

values for all parameters.

It is interesting to note the technique that each interpolation algorithm employs to handle three channel color images and observe if the technique effects the quality of the interpolated result. The Matlab implementation of bicubic interpolation operates on each channel separately. In contrast, edge-directed interpolation works with all color channels simultaneously. Isotropic extensor and adaptive bilateral extensor also handle the color channels together for processing. Results and comparison of the ABE interpolation algorithm are now presented in the following.

Table 5.1: Parameter values used for extensor-based interpolation methods

| Parameter | Isotropic Extensor (IE) | Adaptive Bilateral Extensor (ABE) |
|---------------|-------------------------|---------------------------------------|
| Extensor Size | 7x7 | Adaptive or 5x5 |
| Overlapping | Yes | Yes |
| τ_{edge} | N/A | Adaptive (λ_1, λ_2) |
| σ_i | N/A | Automatic (Eq 4.15) |
| σ_d | N/A | Automatic (Adaptive Structure Tensor) |

Analysis begins utilizing the tire region of the house image. As shown in Figures 5.5-5.8, the ABE method prevents aliasing along edge boundaries, whereas the bicu-

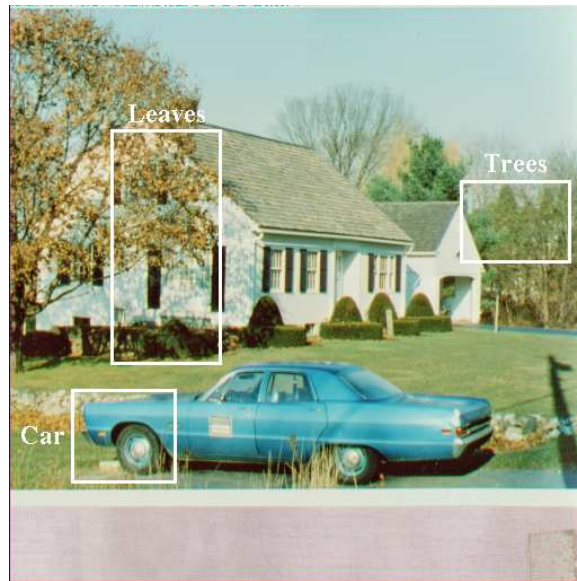


Fig. 5.2: Original house image with regions of interest as labeled.



Fig. 5.3: Decimated tire region of house image.



Fig. 5.4: Original tire region of house image.



Fig. 5.5: Bicubic interpolation by factor two.



Fig. 5.6: IE interpolation by factor two.



Fig. 5.7: Edge-Directed interpolation by factor two.



Fig. 5.8: ABE interpolation by factor two.



Fig. 5.9: Decimated leaves region.



Fig. 5.10: Original leaves region.

bic, edge-directed, and IE algorithms suffer from significant aliasing. An analysis of the SSIM and SNR numerical metrics reveals that the ABE is superior to that of the former algorithms as shown with other images in Tables 5.2 and 5.3. All statistics are computed with a decimated image interpolated by a factor of two and subsequently compared to the original image.

Consideration of the leaves region at a zoom factor of four is shown in Figures 5.11-5.14. Patterning is present in the tree leaves in the bicubic and edge-directed images. Significant artifacts are present in the edge-directed image, almost as though it is an oil painting rather than an interpolated result. The ABE interpolation algorithm produces no patterning or artifacts for a visually superior result to that of the other algorithms. To complete analysis of the house image, Figures 5.17-5.20 illustrate the trees region at a zoom factor of eight. The bicubic and IE interpolation algorithms preserve the trees satisfactorily, however, aliasing is present in the jagged edge of the roof shown in the left portion of the image. The edge-directed method produces a smooth roof line, but creates swirling artifacts in the trees while the ABE algorithm maintains the detail of the trees and produces a smooth roof line, free of aliasing for an improved result as compared to the other algorithms.



Fig. 5.11: Bicubic interpolation by factor four.



Fig. 5.12: IE interpolation by factor four.



Fig. 5.13: Edge-directed interpolation by factor four.



Fig. 5.14: ABE interpolation by factor four.



Fig. 5.15: Decimated trees region.



Fig. 5.16: Original trees region.



Fig. 5.17: Bicubic interpolation by factor eight.



Fig. 5.18: IE interpolation by factor eight.



Fig. 5.19: Edge-Directed interpolation by factor eight.



Fig. 5.20: ABE interpolation by factor eight.



Fig. 5.21: Original airplane (F-16) image with region of interest as labeled.

The airplane (F-16) image is next compared in the text region shown in Figure 5.21 to compare interpolation algorithms on the fine details of written text. Figures 5.22 and 5.23 depict the decimated and original images respectively. To obtain an accurate assessment of the compared interpolation algorithms, the selected region is evaluated at a zoom factor of eight, the results obtained are in Figures 5.24-5.27. The bicubic method produces blocky text and heavy aliasing along the wing of the plane. The IE algorithm produces aliasing around the text in the resulting image and the edge-directed method produces swirling artifacts not present in the original image. However, the ABE interpolation algorithm maintains the original text, free of artifacts and aliasing providing for a visually more pleasing result.



Fig. 5.22: Decimated airplane (F-16) image.



Fig. 5.23: Original airplane (F-16) image.



Fig. 5.24: Bicubic interpolation by factor eight.



Fig. 5.25: IE interpolation by factor eight.



Fig. 5.26: Edge-Directed interpolation by factor eight.



Fig. 5.27: ABE interpolation by factor eight.



Fig. 5.28: Decimated red brick house image.

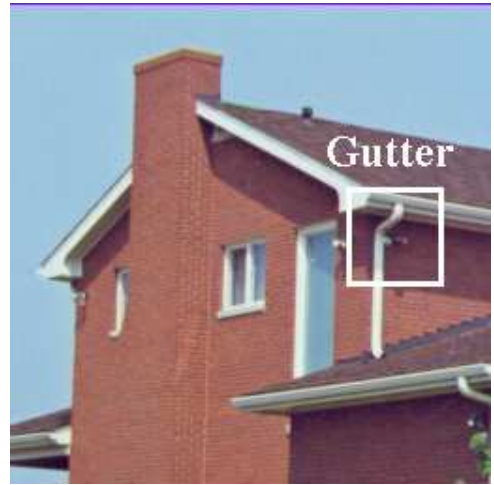


Fig. 5.29: Original sampling of red brick house image.

The red brick house image is utilized for comparison in this thesis for its sharp edge boundaries and unique textures to provide additional facets of comparison between interpolation techniques. The decimated image is shown in Figure 5.28 adjacent to the original in Figure 5.29 with the region further considered as labeled. Evaluation proceeds with the gutter region. The decimated image is shown in figure 5.30 and original in Figure 5.31. The region is interpolated by a factor of eight as shown in Figures 5.32-5.35. Swirling artifacts are clearly visible in the edge-directed image, especially near the top of the gutter drain, whereas the other interpolation methods do not introduce new information to the image. Aliasing is present in the bicubic and IE methods, whereas the ABE interpolation algorithm produces a smooth, well-defined result.



Fig. 5.30: Decimated image of pipe region.

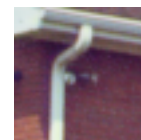


Fig. 5.31: Original image of pipe region.



Fig. 5.32: Bicubic interpolation of pipe region by factor eight.



Fig. 5.33: IE interpolation of pipe region by factor eight.



Fig. 5.34: Edge-Directed image of pipe region at factor eight.



Fig. 5.35: ABE Interpolation of pipe region at factor eight.



Fig. 5.36: Region of remote sensing image at 2.4 meters per pixel.

Remote sensing data are utilized in numerous applications and often require interpolation. Figure 5.36 shows a region of a MS image at 2.4 meters per pixel employed for interpolation with regions used for further analysis as denoted. Results of interpolation techniques are illustrated in Figures 5.39-5.42 for upsampling by a factor of eight, which is employed to readily discern the differences between interpolation algorithms. The source image employed for the house region is shown in Figure 5.37. The bicubic method appears to result in crossing patterns near the edges of shapes in the image providing an irregular patterning while the IE method results in irregular object boundaries most noticeable when viewing the pool. The edge-directed interpolation results in an oil painting look to the result where artifacts are introduced to the image along object boundaries and are not an accurate representation of the source image. The ABE is slightly smoother than bicubic and IE methods, providing a uniform boundary to objects without aliasing or artifacts. Focusing on the pools region at zoom factor eight, original image in Figure 5.38 and results shown in Figures 5.43-5.46, reveals somewhat noticeable aliasing in the bicubic and IE methods.

Moreover, the edge-directed image is clearly swirling edge boundaries and creating new information not present in the original. The ABE image maintains the overall structure of the image while preventing aliasing and maintaining image authenticity.



Fig. 5.37: Original house region.

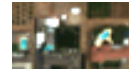


Fig. 5.38: Original pools region.



Fig. 5.39: Bicubic interpolation by factor four.

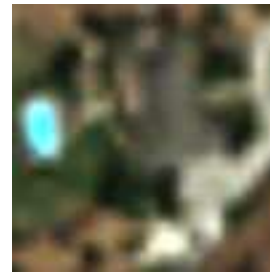


Fig. 5.40: IE interpolation by factor four.



Fig. 5.41: Edge-directed image by factor four.

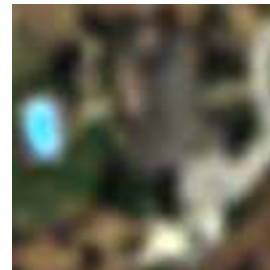


Fig. 5.42: ABE interpolation by factor four.



Fig. 5.43: Bicubic interpolation by factor eight.



Fig. 5.44: IE Interpolation by factor eight.



Fig. 5.45: Edge-directed image by factor eight.



Fig. 5.46: ABE interpolation by factor eight.

Table 5.2: Comparison of SSIM for selected interpolation algorithms. The decimated image is interpolated by a factor of two and numerically compared to the original image as a percent of similarity.















| Image | Filename | ABE | Edge Directed | IE | Bicubic |
|---|----------|--------|---------------|--------|---------|
|  | 4.2.04 | 80.56% | 78.53% | 78.32% | 79.29% |
|  | 4.1.05 | 77.25% | 76.32% | 74.58% | 75.54% |
|  | 4.2.05 | 81.55% | 80.76% | 80.92% | 80.44% |
|  | 4.2.03 | 48.00% | 45.73% | 46.10% | 45.90% |
|  | 4.2.02 | 73.81% | 71.36% | 72.00% | 71.09% |
|  | boat.512 | 66.07% | 65.68% | 64.37% | 65.74% |
|  | 5.2.08 | 69.02% | 68.64% | 67.35% | 68.57% |
|  | 4.1.01 | 83.84% | 82.78% | 82.94% | 82.48% |
|  | 4.1.08 | 89.49% | 88.99% | 89.19% | 88.73% |
|  | house | 72.92% | 71.69% | 71.69% | 71.30% |
|  | 2.1.01 | 54.84% | 54.52% | 53.77% | 53.55% |
|  | 2.1.02 | 46.57% | 45.38% | 45.04% | 45.22% |
|  | 4.2.06 | 70.99% | 70.67% | 70.02% | 68.87% |
|  | 4.1.06 | 71.45% | 71.09% | 70.56% | 70.06% |

Table 5.3: Comparison of SNR for selected interpolation algorithms. The decimated image is upsampled by a factor of two and compared to the original image.




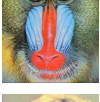
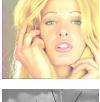



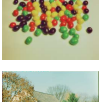



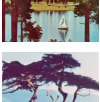
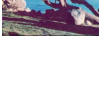
| Image | Filename | ABE | Edge Directed | IE | Bicubic |
|---|----------|---------|---------------|---------|---------|
|  | 4.2.04 | 21.3898 | 20.7823 | 20.5550 | 19.1551 |
|  | 4.1.05 | 20.3910 | 20.1531 | 19.7802 | 17.9652 |
|  | 4.2.05 | 22.0037 | 21.8095 | 21.7618 | 20.1303 |
|  | 4.2.03 | 14.9326 | 14.3525 | 14.4194 | 13.9720 |
|  | 4.2.02 | 25.4642 | 25.1693 | 25.1009 | 22.0406 |
|  | boat.512 | 19.0125 | 14.9894 | 14.8407 | 14.5656 |
|  | 5.2.08 | 15.2723 | 13.5923 | 13.1822 | 12.7825 |
|  | 4.1.01 | 16.5395 | 16.3440 | 16.2465 | 14.2055 |
|  | 4.1.08 | 23.0329 | 22.6778 | 22.7898 | 19.1551 |
|  | house | 19.9427 | 19.6792 | 19.5957 | 19.2263 |
|  | 2.1.01 | 16.6905 | 16.3954 | 16.3170 | 15.6062 |
|  | 2.1.02 | 15.9903 | 15.4980 | 15.5083 | 14.7383 |
|  | 4.2.06 | 18.7727 | 18.5180 | 18.4471 | 17.3961 |
|  | 4.1.06 | 16.4644 | 16.3559 | 16.1978 | 15.0873 |



Fig. 5.47: Region of decimated image for regularization.

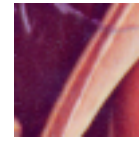


Fig. 5.48: Region of original image for regularization.

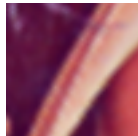


Fig. 5.49: Interpolation factor two with regularization.

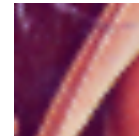


Fig. 5.50: Interpolation factor two without regularization.

5.3 Effect of Regularization

Regularization is utilized in the ABE interpolation algorithm in order to prevent a *string of pearls* that extends along edge boundaries in an image. This observation is most noticeable at high interpolation factors. The ABE algorithm corrects this via regularization of the original pixel values as described in Section 4.4. To clearly illustrate this issue, the hat brim region of Lena is interpolated by a factor of two. Next, nearest neighbor interpolation is employed to further increase visibility. Figure 5.47 depicts the decimated image and Figure 5.48 shows the original. Next, Figures 5.49 and 5.50 illustrate supersampling by factor two with and without regularization, respectively. In addition, the interpolated images are then supersampled by an additional factor of four using nearest neighbor interpolation to allow ease in visualizing the difference provided by regularization. The pearls are due to utilization of original pixel values from the decimated image in the interpolated result. The regularization step ensures that these original pixels are blended smoothly with their surrounding values, providing a uniform appearance to the resulting image.



Fig. 5.51: Interpolation by zoom factor two with regularization, nearest neighbor interpolation by additional factor two.



Fig. 5.52: Interpolation by zoom factor two, without regularization, nearest neighbor interpolation by additional factor two.

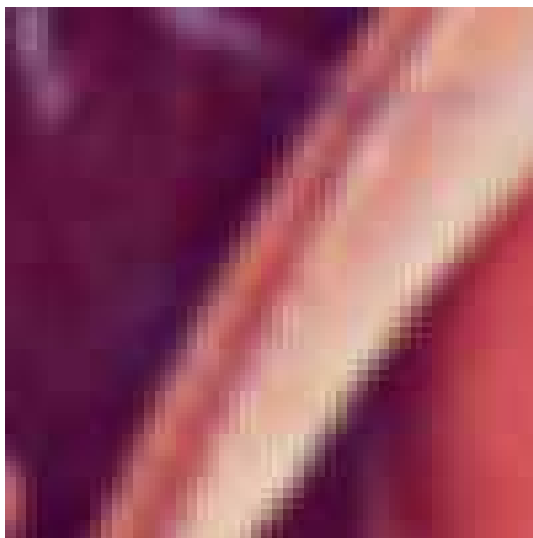


Fig. 5.53: Interpolation by factor four with regularization, nearest neighbor upsampling by additional factor four.

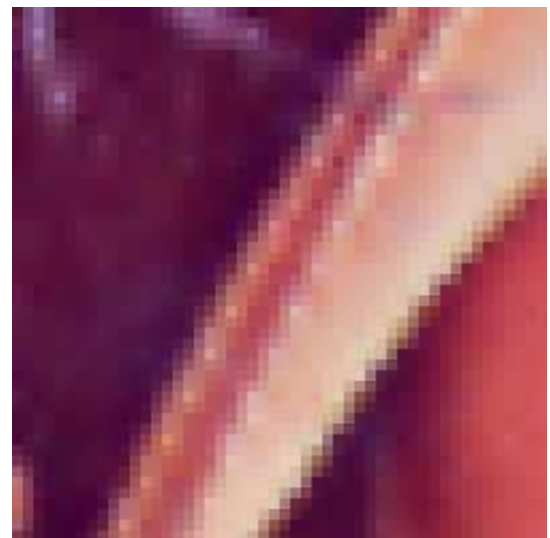


Fig. 5.54: Interpolation by factor four without regularization, nearest neighbor upsampling by additional factor four.

Chapter 6

Pan Sharpening for Multi-Resolution Image Data Sets

Overview

The ABE interpolation algorithm is applied for pan-sharpening of multi-resolution remote sensing data in this chapter. An introduction to remote sensing data is first presented. It is followed by results of pan-sharpening remote sensing images compared with other image interpolation methods utilized for this application.

6.1 Multi-Resolution Remote Sensing Imagery

SATELLITE imagery is increasingly available to businesses, governments, and individuals. Several prominent satellites are actively capturing data including IKONOS, Landsat, and GOES. Each satellite is equipped with different image capture capabilities. The Landsat 7 satellite employs the Enhanced Thematic Mapper Plus (ETM+) allowing for signal capture in eight designated sensor bands. Each sensor band captures a specific wavelength at a designated resolution. Two prominent

types of remote sensing data are MS and Pan. MS data contain several individual spectral bands, often RGB and IR. Pan images capture throughout the RGB and IR wavelengths, but handle the data as a single channel. The Quickbird satellite has maximum resolution capabilities, for commercial applications, of MS data at 2.4 meters per pixel and Pan data at 0.6 meters per pixel [6]. While the resolution of the Pan data is preferable, it is only available in grayscale. Ideally, one would like the MS color present in the 2.4m data combined with the resolution of the Pan data. This process, known as pan-sharpening, aims to provide RGB color to the higher resolution Pan data. Figures 6.1 - 6.4 show each of the four channels of the MS image employed for pan-sharpening. The combined RGB image is shown in Figure 6.5 and a scaled version of the Pan image is in Figure 6.6.



Fig. 6.1: Red channel of the remote sensing region.



Fig. 6.2: Green channel of the remote sensing region.



Fig. 6.3: Blue channel of the remote sensing region.



Fig. 6.4: Near Infrared (IR) channel of the remote sensing region.



Fig. 6.5: Combined RGB channels of remote sensing region.



Fig. 6.6: Scaled image of the panchromatic channel of the remote sensing region.

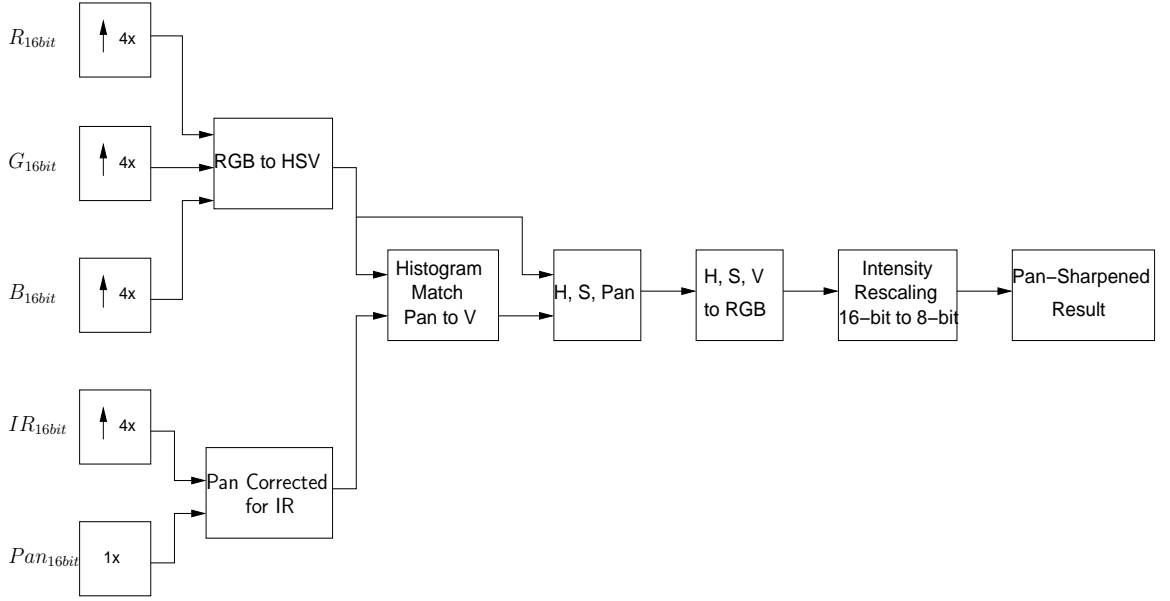


Fig. 6.7: Pan-sharpening algorithm.

The pan-sharpening algorithm is implemented as depicted in Figure 6.7. The MS image is first interpolated by a factor of four to match the resolution of the Pan image. The MS image is next converted from RGB color space to the HSV (Hue, Saturation, Value) color space. HSV provides for a representation of color that will enable fusion of the multi-resolution images via channel replacement. The next step is to correct the Pan image to account for the MS image only containing the red, green, and blue channels after conversion to HSV color space. The IR corrected Pan image, denoted by Pan' is computed as

$$Pan' = Pan - \frac{I_r}{\sum_{j=1}^b MS_j} Pan \quad (6.1)$$

The MS and Pan images may have differences in their histograms, possibly resulting in an incorrect fusion of colors in the result. Histogram matching consists of transforming the Pan histogram to match the value channel of the MS image. The result is a Pan image whose value channel matches that of the MS image, ensuring that the

resulting image colors are blended correctly. The next step is replacement of the MS value channel with that of the Pan corrected channel. The data is then converted back to the RGB color space. Remote sensing data are often stored as 16-bit images. After channel replacement is completed, the 16-bit image is scaled to 8-bit for display. Scaling proceeds by computing the histogram of the image. Next, the upper and lower bounds of the scale are set to 1 percent from the top and bottom of the histogram, respectively. The scaling factor is then computed as

$$I_{8bit} = \frac{(I_{16bit} - I_{min})}{(H_{99\%} - H_{1\%})} \times 255 \quad (6.2)$$

where $H_{99\%}$ and $H_{1\%}$ are the one percent upper and lower bounds of the histogram for a channel. Each channel of data is scaled independently to account for differences based on the level of exposure and variation within the image.

6.2 Pan-Sharpener Results

Results of pan-sharpening using the algorithm detailed in the previous section. The MS image is first interpolated by a factor of four using the current best-performing interpolation algorithms. Shown in Figures 6.11 - 6.14, and original orchard region as labeled in Figure 6.8, the bicubic and IE algorithms result in aliasing along edge boundaries. The edge-directed image produces new artifacts not present in the original image, and the ABE produces a visually accurate result without aliasing or artifacts. To provide a numerical comparison of results, Tables 6.1 and 6.2 list the SNR and SSIM numerical results of the Pan and MS images. Before comparison, the original 16-bit image is decimated by a factor of two. The decimated image is then interpolated by a factor of two and scaled to 8-bit. The original 8-bit scaled image is then compared to the interpolated images.



Fig. 6.8: Region of remote sensing image at 2.4m resolution with regions of interest as labeled. The image is scaled for display, smaller than the original size image.

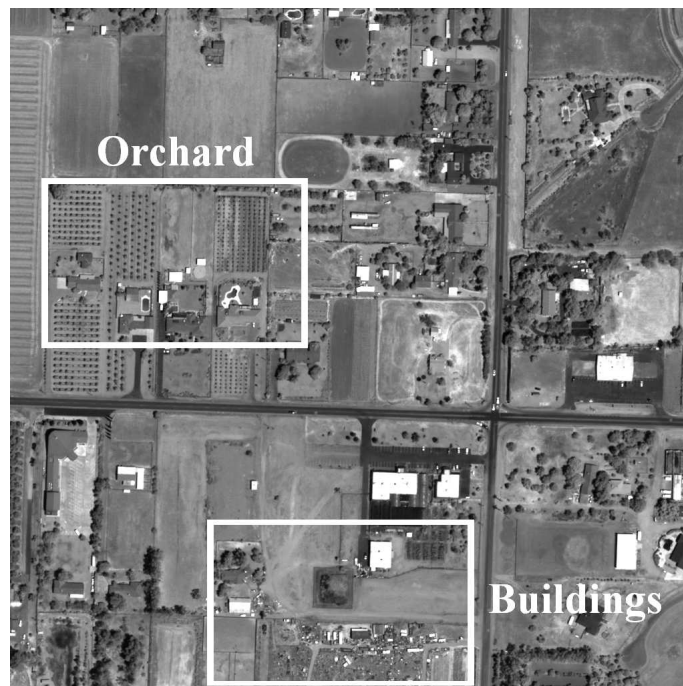


Fig. 6.9: Region of 0.6m Pan image with regions further utilized as labeled, scaled for display corresponding to the 2.4m data.



Fig. 6.10: Original region of the orchard in the MS image



Fig. 6.11: Bicubic interpolation of the orchard region at zoom factor four.



Fig. 6.12: IE interpolation of the orchard region at zoom factor four.



Fig. 6.13: Edge-directed interpolation of the orchard region at zoom factor four.



Fig. 6.14: ABE interpolation of the orchard region at zoom factor four.

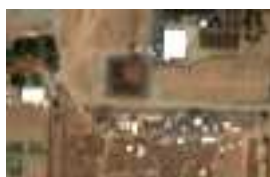


Fig. 6.15: Original region of the buildings in the MS image



Fig. 6.16: Bicubic interpolation of the buildings region at zoom factor four.



Fig. 6.17: IE interpolation of the buildings region at zoom factor four.



Fig. 6.18: Edge-directed interpolation of the buildings region at zoom factor four.



Fig. 6.19: ABE interpolation of the buildings region at zoom factor four.

Table 6.1: Comparison of SNR for selected interpolation algorithms using the remote sensing images. The image is decimated by a factor of two, then upsampled by a factor of two and compared to the original image.





| Image | ABE | Edge Directed | IE | Bicubic |
|---|---------|---------------|---------|---------|
|  | 13.5420 | 13.4183 | 13.3088 | 14.6476 |
|  | 17.3041 | 10.4578 | 17.2921 | 08.6071 |

Table 6.2: Comparison of SSIM for selected interpolation algorithms using the remote sensing images. The image is decimated by a factor of two, then upsampled by a factor of two and compared to the original image.

| Image | ABE | Edge Directed | IE | Bicubic |
|---|--------|---------------|--------|---------|
|  | 64.38% | 62.72% | 62.80% | 80.22% |
|  | 73.82% | 69.08% | 73.97% | 67.17% |

The Pan image is next fused with the interpolated MS data. The source MS and Pan regions are shown in figures 6.10 and 6.20. Results are first shown of the orchard region in figures 6.21-6.24 and original region in figure 6.10. The resulting images appear visually equivalent, contrasting the results shown in figures 6.11 - 6.19 where differences between the interpolation methods are clearly visible. Results obtained

from the second region of interest are depicted in Figures 6.26 - 6.29 with original regions shown in figures 6.15 and 6.25.



Fig. 6.20: Original region of the orchard in the Pan image



Fig. 6.21: Pan-sharpening of orchard region using bicubic interpolation for upsampling the MS image.



Fig. 6.22: Pan-sharpening of orchard region using IE algorithm for upsampling the MS image.



Fig. 6.23: Pan-sharpening of orchard region using edge-directed interpolation for upsampling the MS image.



Fig. 6.24: Pan-sharpening of orchard region using the ABE algorithm for upsampling the MS image.



Fig. 6.25: Original region of the orchard in the Pan image



Fig. 6.26: Pan-sharpening of buildings region using bicubic interpolation for upsampling the MS image.



Fig. 6.27: Pan-sharpening of buildings region using IE algorithm for upsampling the MS image.



Fig. 6.28: Pan-sharpening of buildings region using edge-directed interpolation for upsampling the MS image.



Fig. 6.29: Pan-sharpening of buildings region using the ABE algorithm for upsampling the MS image.

Chapter 7

Conclusion

Overview

A brief overview of algorithms and contributions authored in this thesis are first provided. The results are then summarized, including discussion of current techniques. Application of the ABE to pan-sharpening is also reviewed. Directions of future work and extensions of the current algorithms are discussed with a focus on the future of image interpolation.

7.1 Contributions

DEVELOPMENT of a novel algorithm to solve the image interpolation problem has been presented in this work. It employs the IE method, creating a non-linear mapping from spatial distance to pixel intensity to solve for an unknown pixel value. To refine the extensor window, an adaptive robust structure tensor is utilized to find edge and orientation information and thereby focus the extensor window on nearby pixels, oriented along the edge boundary. These are combined with an edge-preserving bilateral filter to prevent interpolation across edge boundaries, which decreases aliasing and blurring effects, to create a novel method known as Adaptive

Bilateral Extensor for image interpolation.

The ABE interpolation algorithm provides results superior to that of current state-of-the-art methods including bicubic and edge-directed techniques based on qualitative and numerical evaluations metrics. The ABE is specifically advantageous because it does not introduce new information or artifacts to the resulting image, while preventing aliasing along edge boundaries. The result is then applied to pan-sharpening of remote sensing data. Pan-sharpening consists of combining supersampled MS with the original Pan image in the HSV color space by replacing the MS value channel with the normalized, histogram matched Pan image. Results are unexpected in that all of the pan-sharpened images produces excellent quality sharpened images, regardless of the aliasing or artifacts present in the supersampled image. This observation evidences the robustness of pan-sharpening of remote sensing data.

7.2 Future Work

The ABE algorithm described in this thesis makes significant progress in the continuing effort to better interpolate digital imagery. A closely related problem is that of image down-sampling by which an input image has its size reduced by a specified factor, while maintaining detail and preventing aliasing. The ABE method in this work could be adapted with vast potential to improve over current state-of-the-art subsampling methods. One of the most significant challenges remaining in interpolation is the suppression of blurriness as the interpolation factors get very large. Future interpolation algorithms will address this issue and countless additional applications. As the technology and prominence of digital imagery continues to increase, new problems will continue to drive more accurate and efficient algorithms with applications beyond image processing into a plethora of other academic disciplines.

Appendix A

Adaptive Bilateral Extensor (ABE) Implementation

```

1: //First pass using isotropic extensor

2: for each pixel to interpolate in the enlarged image  $I_z$  do

3:      $i \leftarrow$  Position of next pixel

4:      $I_z[i] \leftarrow \mathbf{T}\varphi[i] = \tilde{\mathbf{G}}\mathbf{X}^{-1}\varphi[i] + \vec{\mu}$ 

5: end for

6: //First pass through image to compute adaptive structure tensor

7: for each original pixel in the image do

8:     Compute adaptive structure tensor [see Chapter 3] to identify local image
        structure parameters  $[\lambda_1, \lambda_2]$ 

9:     //Estimate local edge strength

10:     $\tau_{edge} \leftarrow \sqrt{\lambda_1 - \lambda_2}$ 

```

```

11: end for

12: //Second pass through image to selectively apply ABE interpolation

13: for each interpolated pixel do

14:     i ← Position of next pixel in Ω

15:     if  $\tau_{edge} \geq 0.001$  then

16:         //Use ABE interpolation near strong image gradients

17:         extensorWidth ←  $\frac{1}{\sqrt{\lambda_2[i]}}$ 

18:         extensorHeight ←  $\frac{1}{\sqrt{\lambda_1[i]}}$ 

19:         //Compute  $\vec{m}$  within the extensor window

20:         for each original pixel i in the extensor window do

21:              $\sigma_i = \frac{1}{2b} \sum_{i=1}^b \left( \max I_{\vec{p}}^{(i)} - \min I_{\vec{p}}^{(i)} \right)$ 

22:              $\vec{m}_{[i]} \leftarrow \mathbf{B}(\vec{x}) = \frac{1}{k(\vec{x})} \sum_{\vec{p} \in \Omega} \mathbf{W}(\vec{x} - \vec{p}, \tau_{edge}) \mathbf{g}(\mathbf{I}_{\vec{p}} - \mathbf{I}_{\vec{x}}, \sigma_i) I_{\vec{p}}$ 

23:         end for

24:          $I_z[i] \leftarrow \mathbf{T}\varphi[i] = \left( \mathbf{I}_{\vec{m}} \tilde{\mathbf{G}} \right)_{\downarrow} \left( \vec{m} \vec{m}^T \mathbf{X} \right)_{\downarrow \rightarrow}^{-1} \left( \mathbf{I}_{\vec{m}} \cdot \varphi[i] \right)_{\rightarrow} + \vec{\mu}$ 

25:     else

26:         //Use isotropic window size

27:         extensorWidth ← 5

28:         extensorHeight ← 5

```

```
29:          $I_z[i] \leftarrow \mathbf{T}\varphi[i] = \tilde{\mathbf{G}}\mathbf{X}^{-1}\varphi[i] + \vec{\mu}$ 
30:     end if
31: end for
32: //Perform regularization on original pixels
33: extensorWidth  $\leftarrow 3$ 
34: extensorHeight  $\leftarrow 3$ 
35: while all original pixels not regularized do
36:     if  $I_i$  is original then
37:          $I_z[i] \leftarrow \mathbf{T}\varphi[i] = \tilde{\mathbf{G}}\mathbf{X}^{-1}\varphi[i] + \vec{\mu}$ 
38:     end if
39: end while
```

References

- [1] S. Park, M. Park, and M. Kang, “Super-resolution image reconstruction - a technical overview,” in *Proc. IEEE Signal Processing Magazine*, vol. 20, May 2003, pp. 21–36.
- [2] H. Tamayama, K. Ito, and T. Nishimura, “Technology trends of high-definition digital still camera systems,” in *Symposium On VLSI Circuits Digest of Technical Papers*. IEEE, June 2002, pp. 100–105.
- [3] R. Jung, “Image sensor technology for beam instrumentation,” in *BIW98, AIP Conference Proceedings*. Springer-Verlag, May 1998, pp. 74–93.
- [4] Y. Zhang, “Understanding image fusion,” in *Photogrammetric Engineering and Remote Sensing*, vol. 70, no. 6, June 2004, pp. 657–661.
- [5] A. Shamshad, W. W. Hussin, and S. M. Sansui, “Comparison of difference data fusion approaches for surface features extraction using quickbird images,” in *Photogrammetric Engineering and Remote Sensing*, 2004, pp. 1075–1083.
- [6] DigitalGlobe, “Quickbird imagery product guide,” Feb. 2006, pp. 6–9.
- [7] “Space Imaging IKONOS Product Guide,” 2004, pp. 2–5.
- [8] A. Garzelli, F. Nencini, L. Alparone, B. Aiazzi, and S. Baronti, “Pan-sharpening of multispectral images: A critical review and comparison,” in *Proc. IEEE IGARSS*, vol. 1. IEEE, Sept. 2004, pp. 81–84.

-
- [9] J. Vrabel, “Multispectral imagery band sharpening study,” in *Photogrammetric Engineering and Remote Sensing*, vol. 62. American Society for Photogrammetry and Remote Sensing, Sept. 1996, pp. 1075–1083.
- [10] C. Pohl, “Tools and methods for fusion of images of different spatial resolution,” in *Photogrammetric Engineering and Remote Sensing*, vol. 32. International Archives of Photogrammetry and Remote Sensing, June 1999, pp. Part 7–4–3.
- [11] X. Li and M. Orchard, “New edge directed interpolation,” in *Proc. IEEE Int. Conf. Image Processing*, vol. 2. IEEE, Sept. 2000, pp. 311–314.
- [12] K. Palaniappan, J. Uhlmann, and D. Li, “Extensor-based image interpolation,” in *Proc. IEEE Int. Conf. Image Processing*, vol. 3. Columbia, MO USA: IEEE, 2003, pp. 945–948.
- [13] A. Jain, *Fundamentals of Digital Image Processing*. Prentice-Hall, 1988, ISBN 0-133-36165-9.
- [14] D. Li, “Thesis: Extensor-based image and video interpolation and inpainting,” vol. 1, July 2004.
- [15] X. Li and M. Orchard, “New edge-directed interpolation,” in *Proc. IEEE Trans. on Image Processing*, vol. 10. IEEE, Oct. 2001, pp. 1521–1527.
- [16] R.C.Gonzalez and R. E. Woods, *Digital Image Processing*. Upper Saddle River, NJ: Prentice-Hall, 2002, ISBN 0-201-18075-8.
- [17] R. Keys, “Cubic convolution interpolation for digital image processing,” in *IEEE Trans. on Acoustics, Speech, and Signal Processing*, vol. 29. IEEE, December 1981, pp. 1153–1160.

-
- [18] Q. Wang and R. Ward, "A new edge-directed image expansion scheme," in *Proc. IEEE Int. Conf. Image Processing*, vol. 3. IEEE, Oct. 2001, pp. 899–902.
- [19] S.Nath and K.Palaniappan, "Adaptive robust structure tensors for orientation estimation and image segmentation," in *Advances in Visual Computing*, G.Bebis and R.Boyle and D.Koracin and B.Parvin, Ed. Berlin, Germany: Springer-Verlag, 2005, vol. 3804, pp. 17–47, ISBN 3-540-30750-8.
- [20] M. Black and A. Rangarajan, "On the unification of line processes, outlier rejection, and robust statistics with applications in early vision," *Intern. J. Comput. Vis.*, vol. 19, no. 1, pp. 57–91, July 1996.
- [21] C. Tomasi and R. Manduchi, "Bilateral filtering for gray and color images," in *Proc. IEEE Int. Conf. on Computer Vision*, vol. 1. IEEE, January 1998, pp. 417–424.
- [22] G. Sapiro and D. Ringach, "Anisotropic diffusion of color images," in *Proc. SPIE*, vol. 471, Nov. 1996, pp. 1582–1586.
- [23] Z. Wang, A. Bovik, and L. Lu, "Why is image quality assessment so important?" in *Proc. IEEE Int. Conf. on Acoustics, Speech, and Signal Processing*, vol. 4. IEEE, May 2002, pp. 3313–3316.
- [24] Z. Wang, A. Bovik, H. Sheikh, and E. Simoncelli, "Image quality assessment: From error visibility to structural similarity," in *IEEE Trans. On Image Processing*, vol. 13. IEEE, April 2004, pp. 600–612.
- [25] N. Damera-Venkata, T. Kite, W. Geisler, B.L.Evans, and A. Bovik, "Image quality assessment based on a degradation model," in *IEEE Transactions on Image Processing*, vol. 9, April 2000, pp. 636–650.

-
- [26] “National Television System Committee Recommendation ITU-R BT.601-4,” 1994, p. 6.
- [27] D.Muresan and T.Parks, “Adaptive, optimal-recovery image interpolation,” in *Proc. IEEE Int. Conf. Acoustics, Speech, and Signal Processing*, vol. 3. IEEE, May 2001, pp. 1949–1952.
- [28] D.D.Muresan and T.Parks, “Adaptive optimal recovery approach to image interpolation,” in *Proc. IEEE Int. Conf. Image Processing*, vol. 3. IEEE, Oct. 2001, pp. 848–851.
- [29] X.Li, “Source Code for Edge-Directed Interpolation,” [Online] Available: <http://www.csee.wvu.edu/xinl/software.html>.
- [30] Z. Wang, “Demo Images and Free Software for Universal Image Quality Index,” [Online] Available: <http://www.cns.nyu.edu/lev/ssim/>.
- [31] University of Southern California Signal and Image Processing Repository, Images used for results. [Online] Available: <http://sipi.usc.edu/database/>.