Adaptive Image Processing Using Computational Intelligence Techniques

Hau San Wong

School of Electrical and Information Engineering
The University of Sydney
Sydney, NSW 2006, Australia.

A thesis submitted in fulfillment
of the requirements for the
degree of Doctor of Philosophy

December, 1998

Abstract

In this thesis, we illustrate the essential aspects of the adaptive image processing problem in terms of two applications: the adaptive assignment of the regularization parameters in image restoration, and the adaptive characterization of edges in feature detection applications. These two problems are representative of the general adaptive image processing paradigm in that the three requirements for its successive implementation: namely the *segmentation* of an image into its main feature types, the *characterization* of each of these features, and the *optimization* of the image model parameters corresponding to the individual features, are present. In view of these requirements, we have adopted the three main approaches within the class of *computational intelligence* algorithms, namely neural network techniques, fuzzy set theory, and evolutionary computation, for solving the adaptive image processing problem. This is in view of the direct correspondence between some of the above requirements with the particular capabilities of specific computational intelligence approaches.

We first applied neural network techniques to the adaptive regularization problem in image restoration. Instead of the usual approach of selecting the regularization parameter values by trial and error, we adopt a learning approach by treating the parameters in various local image regions as network weights of a model-based neural network with hierarchical architecture (HMBNN), such that they are adjustable through the supply of training examples specifying the desired image quality.

In addition, we also applied the HMBNN to the problem of edge characterization . The representation of the *edge prototypes* as network weights allow their automatic adjustment using a learning process. As a result, instead of communicating their preferences through the specification of thresholds to edge detection algorithms, human users can now highlight specific edge examples on a particular image, and the corresponding pixel configurations are then incorporated as training examples for the neural network.

Returning to the adaptive regularization problem, it was observed that, in addition to the usual requirement of distinguishing low frequency image regions from high frequency

image regions and apply different regularization parameter values to each of them, it is also desirable to further separate the edges and textured areas in the high frequency image regions and regularize them differently due to their unequal noise masking capabilities. This motivates the development of the Edge-Texture Characterization (ETC) measure, which achieves this very purpose by characterizing the correlational properties of local pixel configurations in terms of a scalar value.

The capability of this measure to distinguish edges from textures, together with the inherent ambiguities associated with the terms "edge" and "texture", naturally suggests the formulation of a fuzzy model where these two concepts are represented as two fuzzy sets defined on the domain of ETC measures. At the same time, the previous sub-network structure of the HMBNN is extended to include two neurons which respectively computes the associated restoration variables for edges and textures . The output of these two neurons are then combined in an augmented *output layer* to give the final amount of gray level update required, with the edge/texture fuzzy membership values, which reflects whether the current pixel neighborhood resembles more to edges or textures, as the two output weights.

Apart from using NN and fuzzy techniques, we also investigated the feasibility of applying evolutionary computation techniques to this problem. This is due to our observation of the similar shapes of the histograms formed from the ETC measure values for a large class of images, and the possible deviations from this standard shape for degraded or improperly regularized image. Our purpose, therefore, is to adaptively regularize an image such that its associated ETC-histogram again exhibits the standard shape. The non-differentiability of the associated cost function necessitates the adoption of evolutionary computational techniques which does not depend on the availability of gradient information. An important implication of this EC approach is the possibility to specify an image processing application in terms of a more accurate cost function instead of a sub-optimal but differentiable one.

Acknowledgments

I would like to express my sincere thanks to Dr Ling Guan, my supervisor, for his guidance and encouragement throughout the course of this research. I am also grateful to Prof. Terry Caelli for the many stimulating exchanges which eventually lead to the work in Chapter 4 of this thesis.

I would also like to thank my colleagues in the Signal and Multimedia Processing Group in the University of Sydney, especially Hao Song Kong, Stuart Perry and Songyang Yu, for their help throughout the course of this research.

Publication List

Journal Publications

H. S. Wong, T. Caelli and L. Guan, " A Model-Based Neural Network for Edge Characterization ," submitted to *Pattern Recognition.*

H. S. Wong and L. Guan, " Application of Evolutionary Programming to Adaptive Regularization in Image Restoration," submitted to *IEEE Trans. on Evolutionary Computation* (under revision).

H. S. Wong and L. Guan, " Adaptive Regularization in Image Restoration By Unsupervised Learning," *Journal of Electronic Imaging,*,vol. 7, no.1, pp.211-221,1998.

H. S. Wong and L. Guan, " Adaptive Regularization in Image Restoration Using a Model-Based Neural Network," *Optical Engineering*, vol.36 no.12, pp.3297-3308, 1997.

Conference Publications

H. S. Wong, T. Caelli and L. Guan, "Edge Characterization Using a Model-Based Neural Network," to be presented in *1999 IEEE Int. Conf. on Acoustics, Speech and Signal Processing.*

H. S. Wong and L. Guan, "Adaptive Regularization in Image Restoration Using Evolutionary Programming," *Proc. IEEE Int. Conf. on Evolutionary Computation*, pp.159-164, 1998.

L. Guan, S. Perry, R. Romagnoli, H. S. Wong and H. S. Kong, "Neural Vision System and Applications in Image Processing and Analysis," *Proc. IEEE Int. Conf. on Acoustics, Speech and Signal Processing,* pp.1245-1248, 1998.

H. S. Wong and L. Guan, " Adaptive Regularization in Image Restoration Using a Model-Based Neural Network,"in *Applications of Artificial Neural Network in Image processing II, N.M. Nasrabadi, A.K. Katsaggelos,eds. Proc. SPIE 3030*, pp.125-136, 1997.

H. S. Wong and L. Guan, " A Study of Adaptive Regularization for Image Restoration," in *Proc. Int. Conf. on Control, Automation, Robotics and Computer Vision*, pp.665-669, 1996.

H. S. Wong, L. Guan and H. Kong, "Compression of Digital Mammogram Databases Using a Near-Lossless Scheme," *Proc. IEEE Int. Conf. on Image Processing*, pp.21-24, 1995.

Contents

1 Introduction **1**

1.1 Introduction 1

1.1.1 The Need for Adaptive Image Processing 1

1.1.2 The Three Main Image Feature Classes 2

1.1.3 Difficulties in Adaptive Image Processing System Design 4

1.1.4 Computational Intelligence Techniques 7

1.2 Scope of the Thesis 14

1.2.1 Adaptive Regularization 14

1.2.2 Edge Characterization and Detection 16

1.3 Contributions of the Current Work 17

1.3.1 Application of Model-Based Neural Networks to Adaptive Regularization 17

1.3.2 Application of Model-Based Neural Networks to Edge Characterization 18

1.3.3 Formulation of the Edge-Texture Characterization (ETC) Measure . 19

1.3.4 Incorporation of the ETC Measure in a Fuzzy Model-Based NN for Adaptive Regularization 20

1.3.5 Application of Evolutionary Programming to Adaptive Regularization 20

1.4 Organization of the Thesis 21

2 Model-Based Neural Network with Hierarchical Architecture (HMBNN) 23

2.1 Introduction 23

2.1.1 Supervised Learning Algorithms 25
2.1.2 Unsupervised Learning Algorithms 26
2.2 Model-Based Neural Network 26
2.2.1 Weight-parameterized model-based neuron 27
2.2.2 Input-parameterized model-based neuron 28
2.3 Hierarchical Neural Network Architecture 30
2.3.1 Hidden-Node Hierarchical Architecture 31
2.3.2 Subcluster Hierarchical Architecture 33
2.3.3 Combination of Sub-Network Outputs 33
2.4 Model-Based Neural Network with Hierarchical Architecture (HMBNN) . . 35
2.5 HMBNN for Adaptive Image Processing 36
2.5.1 Adaptive Regularization in Image Restoration 36
2.5.2 Edge Characterization and Detection 37

3 Application of HMBNN to Adaptive Regularization **39**

3.1 Introduction 39
3.2 The Hopfield Neural Network Model for Image Restoration 42
3.2.1 Optimization of the Restoration Cost Function 42
3.3 Adaptive Regularization–An Alternative Formulation 45
3.3.1 Correspondence with the General HMBNN Architecture 47
3.4 Regional Training Set Definition 51
3.5 Determination of the Image Partition 55
3.6 Experimental Results 57
3.7 Summary 74

4 Application of HMBNN to Edge Characterization **76**

4.1 Introduction 76
4.2 Network Architecture 78
4.2.1 Characterization of Edge Information 78
4.2.2 Sub-Network G_r 79

4.2.3 Neuron N_{rs} in Sub-Network G_r 80
4.2.4 Dynamic tracking neuron N_d 81
4.2.5 Binary edge configuration 82
4.2.6 Correspondence with the General HMBNN architecture 83
4.3 Network Training 85
4.3.1 Determination of g_r for sub-network G_r 85
4.3.2 Determination of $\mathbf{w}_{rs}$ for neuron N_{rs} 85
4.3.3 Acquisition of valid edge configurations 86
4.4 Recognition Phase 86
4.4.1 Location of primary edge points 87
4.4.2 Location of secondary edge points 87
4.4.3 Determination of the network size 90
4.5 Experimental Results 91
4.6 Summary 103

5 Distinguishing Between Edge and Texture: the Edge-Texture Characterization (ETC) Measure **107**

5.1 Introduction 107
5.2 The Edge-Texture Characterization (ETC) Measure 109
5.3 Experimental Results 112
5.4 An algorithm for textured area extraction 117
5.4.1 The Texture Extraction Algorithm 117
5.5 Experimental Results 119
5.6 Summary 123

6 The ETC Fuzzy HMBNN for Adaptive Regularization **124**

6.1 Theory of Fuzzy Sets 125
6.2 Edge-Texture Fuzzy Model Based on ETC measure 128
6.3 Architecture of the Fuzzy HMBNN 130
6.3.1 Correspondence with the General HMBNN Architecture 132

6.4 Estimation of the Desired Network Output 132
6.5 Fuzzy Prediction of Desired Gray Level Value 134
6.5.1 Definition of the fuzzy estimator membership function 134
6.5.2 Fuzzy Inference Procedure for Predicted Gray Level Value 136
6.5.3 Defuzzification of the fuzzy set G 137
6.5.4 Regularization Parameter Update 139
6.5.5 Update of the Estimator Fuzzy Set Width Parameters 140
6.5.6 Incorporation of the Texture Map 142
6.5.7 Experimental Results 144
6.6 Summary 161

7 Application of Evolutionary Programming to Adaptive Regularization 163
7.1 Introduction 163
7.2 Introduction to Evolutionary Computation 164
7.2.1 Genetic Algorithm (GA) 165
7.2.2 Evolutionary Strategy (ES) 166
7.2.3 Evolutionary Programming (EP) 168
7.3 The ETC-pdf Image Model 169
7.4 Adaptive Regularization Using Evolutionary Programming 175
7.4.1 Competition Under Approximate Fitness Criterion 179
7.4.2 Choice of optimal regularization strategy 181
7.5 Incorporation of ETC Fuzzy Criterion 184
7.6 Experimental results 186
7.7 Summary 204

8 Conclusions 206
8.1 Main Conclusions 206
8.1.1 Effectiveness of adopting a learning approach in adaptive regularization 208
8.1.2 Robustness of neural network-based edge characterization 209

8.1.3 Formulation of the Edge-Texture Characterization (ETC) measure 210
8.1.4 Incorporation of edge-texture discrimination in adaptive regularization 210
8.1.5 Effectiveness of evolutionary parameterization in image restoration 211
8.2 Suggestions for Future Works 212
8.2.1 Improvements for Current Works 212
8.2.2 Extensions to other application areas 214

List of Figures

1.1 The three important classes of feature in images 2

1.2 Examples of edge models 3

1.3 The three main requirements in adaptive image processing 7

1.4 The three main classes of computational intelligence algorithms 8

1.5 Relationships between the computational intelligence algorithms and the main requirements in adaptive image processing 13

2.1 The architecture of a neural network with one hidden layer 25

2.2 The weight-parameterized model-based neuron 28

2.3 The input-parameterized model-based neuron 30

2.4 Hidden-node hierarchical network architecture (a) global network architecture (b) sub-network architecture 32

2.5 Subcluster hierarchical network architecture (a) global network architecture (b) sub-network architecture 34

3.1 The model-based neural network with hierarchical architecture for adaptive regularization 48

3.2 The model-based neuron for adaptive regularization 49

3.3 Images used in the adaptive regularization experiments. 58

3.4 Restored flower image using non-fuzzy HMBNN (5×5 Gaussian blur, 30dB BSNR). 60

3.5 Restored flower images using non-fuzzy HMBNN (5×5 uniform blur, 30dB BSNR) 61

3.6 Restored flower images using non-fuzzy HMBNN (5×5 uniform blur, 20dB BSNR). 63
3.7 Evolution of the segmentation map of flower during the restoration process 64
3.8 Evolution of the regional regularization parameters in the flower image during the restoration process. 66
3.9 Restored Lena image using non-fuzzy HMBNN (5×5 uniform blur, 30dB BSNR). 67
3.10 Restored Eagle image using non-fuzzy HMBNN (5×5 uniform blur, 30dB BSNR). 68
3.11 The λ-distribution maps for the three images under 5×5 uniform blur at different levels of additive noise. 69
3.12 Degraded images of Lena and eagle (5×5 uniform blur, 20dB BSNR). . . 71
3.13 Restored images using the edge-oriented and texture-oriented prediction schemes (5×5 uniform blur, 20dB BSNR). 72

4.1 The architecture of the HMBNN edge detector 79
4.2 The architecture of a single sub-network G_r 81
4.3 The structure of the dynamic edge tracking neuron 82
4.4 Examples of valid edge configurations 83
4.5 Illustrating the operation $R_{\frac{\pi}{4}}$ 83
4.6 (a) Eagle image (b)-(e) Edge examples supplied by human user. 93
4.7 (a) Flower image (b)-(e) Edge examples supplied by human user. 94
4.8 Edge tracings by human users in independent trials. 95
4.9 Edge detection results for the eagle image using NN and the Shen-Castan edge detector with different thresholds 96
4.10 Edge detection results for the eagle image using the Shen-Castan edge detector with different filter parameters 97
4.11 Edge detection results for the flower image using NN and the Shen-Castan edge detector with different thresholds 100

4.12 Edge detection results for the flower image using the Shen-Castan edge detector with different filter parameters 101
4.13 Edge detection results for the clock image and a natural scenery image using NN 102
4.14 Edge detection results for the eagle image with additive Gaussian noise, using NN and the Shen-Castan edge detector with different thresholds. . . 104
4.15 NN edge detection results for the flower image, the clock image and a natural scenery image with additive Gaussian noise 105

5.1 Illustrating the various forms of partition $\mathcal{P}$ 109
5.2 ETC distribution map of flower image 114
5.3 ETC distribution map of Lena image 115
5.4 ETC distribution map of eagle image 116
5.5 The two structuring elements used in the texture extraction algorithm . . . 118
5.6 Illustrating the morphological operations for texture extraction 119
5.7 Performance of textured region extraction algorithm 120
5.8 Texture extraction results for degraded images 121

6.1 The fuzzy set representing the concept TALL 126
6.2 The EDGE and TEXTURE fuzzy membership functions 128
6.3 The architecture of the fuzzy HMBNN sub-network 131
6.4 The fuzzy membership functions of the sets EG and TG 135
6.5 The mapping of the fuzzy sets EG and TG to G 137
6.6 Restored flower images using fuzzy HMBNN (5×5 Gaussian blur ($\sigma_g = 1$), 30dB BSNR). 146
6.7 Restored flower images using fuzzy HMBNN (5×5 uniform blur, 30dB BSNR). 147
6.8 Restored flower images using fuzzy HMBNN (5×5 uniform blur, 20dB BSNR). 148

6.9 Restored Lena images using fuzzy HMBNN (5 × 5 uniform blur, 30dB BSNR). 151
6.10 Restored Lena images using fuzzy HMBNN (5 × 5 uniform blur, 20dB BSNR). 152
6.11 Restored eagle images using fuzzy HMBNN (5 × 5 uniform blur, 30dB BSNR). 153
6.12 Restored eagle images using fuzzy HMBNN (5 × 5 uniform blur, 20dB BSNR). 154
6.13 The λ-distribution maps for the three images under 5 ×5 uniform blur at different levels of additive noise. 157
6.14 Edge/texture classification under HMBNN regularization. 158

7.1 The ETC-pdf of different images 170
7.2 Gray level histograms of different images. 172
7.3 The ETC-pdf for a typical image and its blurred version. 173
7.4 Illustrating the various parameters of a typical regularization profile $\mathcal{S}_p$ . . 177
7.5 Illustrating the assignment of regions $\mathcal{R}_p$ to individual regularization strategy $\mathcal{S}_p$ for a 4-member population. 181
7.6 Restored flower images using non-fuzzy EP (5 × 5 Gaussian blur, 30dB BSNR). 188
7.7 Restored flower images using non-fuzzy EP (5 × 5 uniform blur, 30dB BSNR). 189
7.8 Restored flower images using non-fuzzy EP (5 × 5 uniform blur, 20dB BSNR). 190
7.9 Restored Lena images using non-fuzzy EP (5 × 5 uniform blur, 30dB BSNR).192
7.10 Restored eagle images using non-fuzzy EP (5 × 5 uniform blur, 30dB BSNR).193
7.11 The λ-distribution maps for the three images under 5 ×5 uniform blur at different levels of additive noise. 194

7.12 Restored images of Lena and eagle using non-fuzzy EP (5 × 5 uniform blur, 20dB BSNR). 198
7.13 Restored flower images using fuzzy EP (5 × 5 uniform blur, 30dB BSNR). 199
7.14 Restored Flower images using fuzzy EP (5 × 5 uniform blur, 20dB BSNR). 200
7.15 Restored Lena images using fuzzy EP (5 × 5 uniform blur, 20dB BSNR). 202
7.16 Restored eagle images using fuzzy EP (5 × 5 uniform blur, 20dB BSNR). 203

List of Tables

3.1 Average regional regularization parameter values for the two region types . 70

5.1 ETC measure values for various different partitions 111

6.1 RMSE values of the restoration results using various algorithms 156

6.2 Average regional regularization parameter values for the three feature types 159

7.1 Comparison of the final evolved regularization strategy under different degradation conditions 196

7.2 RMSE values of the restoration results using various algorithms 204

Chapter 1

Introduction

1.1 Introduction

1.1.1 The Need for Adaptive Image Processing

The need for adaptive image processing arises from the non-stationary nature of the two dimensional image signals. As opposed to stationary random processes where the statistical properties of the signal remain unchanged with respect to the 2-D spatial index, the very word "image" in an everyday sense almost always implies the presence of inhomogeneity in it. It is this inhomogeneity that conveys useful information of a scene, usually composed of a number of different objects, to the viewer. On the other hand, a stationary 2-D random signal, when viewed as a gray-level image, does not usually correspond to the appearances of real-world objects.

For a particular image processing application (we interpret the term "image processing" in a wide sense such that applications in image analysis are also included), we usually assume the existence of an underlying *image model* [49, 50, 93] , which is a mathematical description of a hypothetical process through which the current image is generated. If we suppose that an image is adequately described by a stationary random process, which, though not accurate in general, is often invoked as a simplifying assumption, it is apparent that only a single image model corresponding to this random process is required

for further image processing. On the other hand, more sophisticated image processing algorithms will account for the non-stationarity of real images by adopting *multiple* image models for more accurate representation. Individual regions in the image can usually be associated with a different image model, and the complete image can be fully characterized by a finite number of these local image models.

1.1.2 The Three Main Image Feature Classes

The inhomogeneity in images implies the existence of more than one image feature type which convey independent forms of information to the viewer. Although variations among different images can be great, a large number of images can be characterized by a small number of feature types. These are usually summarized under the labels of smooth regions, textures and edges (Figure 1.1). In the following, we will describe the essential characteristics of these three kinds of features, and the image models usually employed for their characterization.

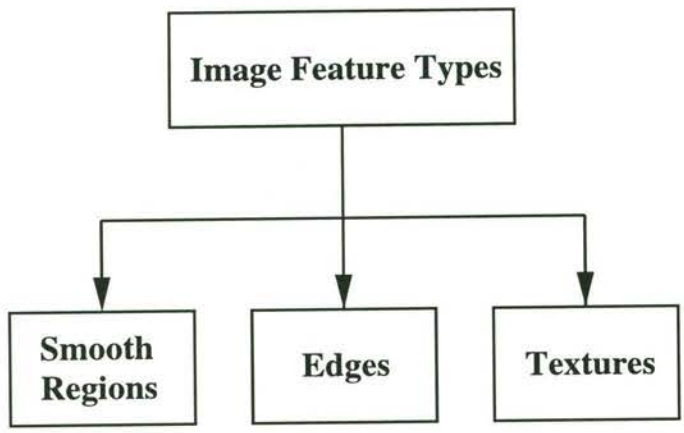


Figure 1.1: The three important classes of feature in images

Smooth regions

Smooth regions usually comprise the largest proportion of areas in images, due to the reason that surfaces of artificial or natural objects, when imaged from a distance, can usually be regarded as smooth. A simple model for a smooth region is the assignment of

a constant gray level value to a restricted domain of the image lattice, together with the addition of Gaussian noise of appropriate variance to model the sensor noise [36, 50].

Edges

As opposed to smooth regions, edges comprise only a very small proportion of areas in images. Nevertheless, most of the information in an image is conveyed through these edges. This is easily seen when we look at the edge map of an image after edge detection: we can readily infer the original contents of the image through the edges alone. Since edges represent locations of abrupt transition of gray level values between adjacent regions , the simplest edge model is therefore a random variable of high variance, as opposed to the smooth region model which use random variables with low variances. However, this simple model does not take into account the structural constraints in edges, which may then lead to their confusion with textured regions with equally high variances. More sophisticated edge models include the facet model [39] , which approximate the different regions of constant gray level values around edges with separate piecewise continuous functions. There are also the edge profile model, which describes the one-dimensional cross section of an edge in the direction of maximum gray level variation [22, 98]. Attempts have been made to model this profile using a step function and various monotonically increasing functions. Whereas these models mainly characterize the *magnitude* of gray level value transition at the edge location, the edge diagram in terms of zero crossings of the second order gray level derivatives, obtained through the process of Laplacian of Gaussian (LoG) filtering [74, 84] , characterizes the edge *positions* in an image. These three edge models are illustrated in Figure 1.2.

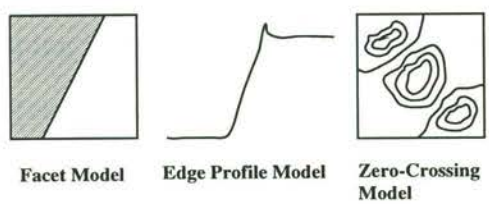


Figure 1.2: Examples of edge models

Textures

The appearance of textures is usually due to the presence of natural objects in an image. The textures usually have a noise-like appearance, although they are distinctly different from noises in that there usually exists certain discernible patterns within them. This is due to the correlations among the pixel values in specific directions. Due to this noise-like appearance, it is natural to model textures using a 2-D random field. The simplest approach is to use i.i.d (independent and identically distributed) random variables with appropriate variances, but this does not take into account the correlations among the pixels. A generalization of this approach is the adoption of Gauss Markov Random Field (GMRF) [23, 24, 26, 54, 112] and Gibbs random field [27, 33] which model these local correlational properties. Another characteristic of textures is their self-similarities: the patterns usually look similar when observed under different magnifications. This leads to their representation as fractal processes [13, 73] which possess this very self-similar property.

1.1.3 Difficulties in Adaptive Image Processing System Design

Given the very different properties of these three feature types, it is usually necessary to incorporate spatial adaptivity into image processing systems for optimal results. For an image processing system, a set of system parameters is usually defined to control the quality of the processed image. Assuming the adoption of spatial domain processing algorithms, the gray level value x_{i_1,i_2} at spatial index (i_1, i_2) is determined according to the following relationship

$$x_{i_1,i_2} = f(\mathbf{y}; \mathbf{p}_{SA}(i_1, i_2)) \tag{1.1}$$

In this equation, the mapping f summarizes the operations performed by the image processing system. The vector $\mathbf{y}$ denotes the gray level values of the original image before processing, and $\mathbf{p}_{SA}$ denotes a vector of *spatially adaptive* parameters as a function of the spatial index (i_1, i_2). It is reasonable to expect that different parameter vectors are to be adopted at different positions (i_1, i_2), which usually correspond to different feature types.

As a result, an important consideration in the design of this adaptive image processing system is the proper determination of the parameter vector $\mathbf{p}_{SA}(i_1, i_2)$ as a function of the spatial index (i_1, i_2).

On the other hand, for non-adaptive image processing systems, we can simply adopt a constant assignment for $\mathbf{p}_{SA}(i_1, i_2)$

$$\mathbf{p}_{SA}(i_1, i_2) \equiv \mathbf{p}_{NA} \tag{1.2}$$

where $\mathbf{p}_{NA}$ is a constant parameter vector.

We consider examples of $\mathbf{p}_{SA}(i_1, i_2)$ in a number of specific image processing applications below.

- In image filtering, we can define $\mathbf{p}_{SA}(i_1, i_2)$ to be the set of filter coefficients in the convolution mask [50] . Adaptive filtering [5, 101] thus corresponds to using a different mask at different spatial locations, while non-adaptive filtering adopts the same mask for the whole image.

- In image restoration [3, 12, 55] , a *regularization parameter* [14, 47, 88] is defined which controls the degree of ill-conditioning of the restoration process, or equivalently, the overall smoothness of the restored image. The vector $\mathbf{p}_{SA}(i_1, i_2)$ in this case corresponds to the scalar regularization parameter. Adaptive regularization [56, 68, 110] involves selecting different parameters at different locations, and non-adaptive regularization adopts a single parameter for the whole image.

- In edge detection, the usual practice is to select a single *threshold* parameter on the gradient magnitude to distinguish between the edge and non-edge points of the image [36, 50] , which corresponds to the case of non-adaptive thresholding. This can be considered as a special case of adaptive thresholding, where a threshold value is defined at each spatial location.

Given the above description of adaptive image processing, we can see that the corresponding problem of adaptive parameterization, that of determining the parameter vector $\mathbf{p}_{SA}(i_1, i_2)$ as a function of (i_1, i_2), is particularly acute compared with the non-adaptive

case. In the non-adaptive case, and in particular for the case of a parameter vector of low dimensionality, it is usually possible to determine the optimal parameters by interactively choosing different parameter vectors and evaluating the final processed results.

On the other hand, for adaptive image processing, it is almost always the case that a parameter vector of high dimensionality, which consists of the concatenation of all the local parameter vectors, will be involved. If we relax the previous requirement to allow the sub-division of an image into regions and the assignment of the same local parameter vector to each region, the dimension of the resulting concatenated parameter vector can still be large. In addition, the requirement to identify each image pixel with a particular feature type itself constitutes a non-trivial *segmentation* problem. As a result, it is usually not possible to estimate the parameter vector by trial and error. Instead, we should look for a parameter assignment algorithm which would automate the whole process.

To achieve this purpose, we will first have to establish image models which describe the desired local gray level value configurations for the respective image feature types, or in other words, to *characterize* each feature type. Since the local gray level configurations of the processed image are in general a function of the system parameters as specified in Eq (1.1), we can associate a *cost function* with each gray level configuration which measures its degree of conformance to the corresponding model, with the local system parameters as arguments of the cost function. We can then search for those system parameter values which minimize the cost function for each feature type, i.e., an *optimization* process. Naturally, we should adopt *different* image models in order to obtain different system parameters for each type of feature.

In view of these requirements, we can summarize the requirements for a successful design of an adaptive image processing system as follows:

Segmentation

Segmentation requires a proper understanding of the difference between the corresponding structural and statistical properties of the various feature types, including those of edges,

textures and smooth regions, to allow partition of an image into these basic feature types.

Characterization

Characterization requires an understanding of the most desirable gray level value configurations in terms of the characteristics of the Human Vision System (HVS) for each of the basic feature types, and the subsequent formulation of these criteria into cost functions in terms of the image model parameters, such that the minimization of these cost functions will result in an approximation to the desired gray level configurations for each feature type.

Optimization

In anticipation of the fact that the above criteria will not necessarily lead to well-behaved cost functions, and that some of the functions will be non-linear or even non-differentiable, we should adopt powerful optimization techniques for the searching of the optimal parameter vector.

These three main requirements are summarized in Figure 1.3.

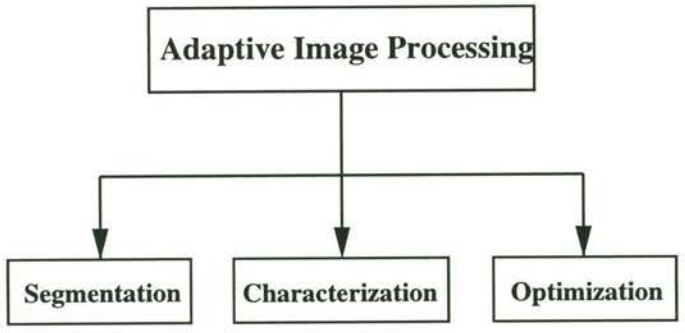


Figure 1.3: The three main requirements in adaptive image processing

1.1.4 Computational Intelligence Techniques

Considering the above stringent requirements for the satisfactory performance of an adaptive image processing systems, it will be natural to consider the class of algorithms com-

monly known as computational intelligence techniques. The term "computational intelligence" [4, 83] has sometimes been used to refer to the general attempt to simulate human intelligence on computers, the so-called "artificial intelligence" (AI) approach [109] . However, in this thesis, we will adopt a more specific definition of computational intelligence techniques according to the three encompassed fields in the IEEE World Congress on Computational Intelligence [89] , which are neural network techniques, fuzzy logic and evolutionary computation (Figure 1.4). These are also referred to as the "numerical" AI approaches (or sometimes "soft computing" approach [51]) in contrast to the "symbolic" AI approaches as typified by the expression of human knowledge in terms of linguistic variables in expert systems [109].

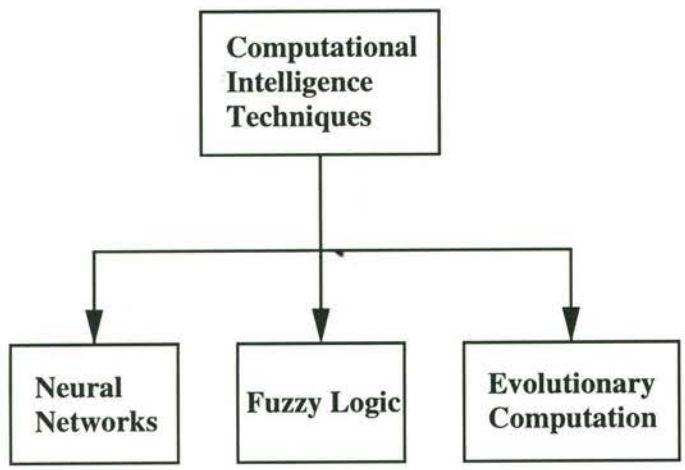


Figure 1.4: The three main classes of computational intelligence algorithms

A distinguishing characteristic of this class of algorithms is that they are usually biologically inspired: the design of neural networks [41, 42] , as the name implies, draws the inspiration mainly from the structure of the human brain. Instead of adopting the serial processing architecture of the Von Neumann computer, a neural network consists of a large number of computational units or neurons (the use of this term again confirming the biological source of inspiration) which are massively interconnected with each other just as the real neurons in the human brain are interconnected with axons and dendrites. Each such connection between the artificial neurons is characterized by an adjustable *weight* which can be modified through a training process such that the overall behavior

of the network is changed according to the nature of specific training examples provided, again reminding one of the human learning process.

On the other hand, fuzzy logic [59, 61, 113] is usually regarded as a formal way to describe how human beings perceive everyday concepts: whereas there is no exact height or speed corresponding to concepts like "tall" and "fast" respectively, there is usually a general consensus by humans as to approximately what levels of height and speed the terms are referring to. To mimic this aspect of human cognition on a machine, fuzzy logic avoids the arbitrary assignment of a particular numerical value to a single class. Instead, it defines each such class as a *fuzzy set* as opposed to a *crisp set*, and assigns a fuzzy set membership value within the interval $[0, 1]$ for each class which expresses the degree of membership of the particular numerical value in the class, thus generalizing the previous concept of crisp set membership values within the discrete set $\{0, 1\}$.

For the third member of the class of computational intelligence algorithms, no concept is closer to biology than the concept of evolution, which is the incremental adaptation process by which living organisms increase their fitness to survive in a hostile environment through the processes of mutation and competition. Central to the process of evolution is the concept of a *population* in which the better adapted individuals gradually displace the not so well adapted ones. Described within the context of an optimization algorithm, an *evolutionary computational algorithm* [7, 29] mimics this aspect of evolution by generating a population of potential solutions to the optimization problem, instead of a sequence of single potential solution as in the case of gradient descent optimization or simulated annealing [33] . The potential solutions are allowed to compete against each other by comparing their respective cost function values associated with the optimization problem with each other. Solutions with high cost function values are displaced from the population while those with low cost values survive into the next generation. The displaced individuals in the population are replaced by generating new individuals from the survived solutions through the processes of mutation and recombination. In this way, many regions in the search space can be explored simultaneously, and the search process is not affected by local minima as no gradient evaluation is required for this algorithm.

We will now have a look at how the specific capabilities of these computational intelligence techniques can address the various problems encountered in the design and parameterization of an adaptive image processing system.

Neural Networks

The adaptive capability of neural networks through the adjustment of the network weights will prove useful in addressing the requirements of segmentation, characterization and optimization in adaptive image processing system design. For segmentation, we can, for example, ask human users to specify which part of an image corresponds to edges, textures and smooth regions, etc. We can then extract image features from the specified regions as training examples for a properly designed neural network such that the trained network will be capable of segmenting a previously unseen image into the primitive feature types. Previous works where neural network is applied to the problem of image segmentation are detailed in [6, 34, 46]

Neural network is also capable of performing characterization to a certain extent, especially in the process of *unsupervised competitive learning* [41, 60], where both segmentation and characterization of training data are carried out: during the competitive learning process, individual neurons in the network, which represent distinct sub-classes of training data, gradually build up *templates* of their associated sub-classes in the form of *weight vectors*. These templates serve to characterize the individual sub-classes.

In anticipation of the possible presence of non-linearity in the cost functions for parameter estimation during the optimization process, neural network is again an ideal candidate for accommodating such difficulties: the operation of a neural network is inherently nonlinear due to the presence of the sigmoid neuronal transfer function. We can also tailor the nonlinear neuronal transfer function specifically to a particular application. More generally, we can *map* a cost function onto a neural network by adopting an architecture such that the image model parameters will appear as adjustable weights in the network [86, 118] . We can then search for the optimal image model parameters by

minimizing the embedded cost function through the dynamic action of the neural network.

In addition, while the distributed nature of information storage in neural networks and the resulting fault-tolerance is usually regarded as an overriding factor in its adoption, we will, in this thesis, concentrate rather on the possibility of task localization in a neural network: we will sub-divide the neurons into *neuron clusters*, with each cluster specialized for the performance of a certain task [1, 65] . It is well known that similar localization of processing occurs in the human brain, as in the classification of the cerebral cortex into visual area, auditory area, speech area and motor area, etc [52, 102]. In the context of adaptive image processing, we can, for example, subdivide the set of neurons in such a way that each cluster will process the three primitive feature types, namely textures, edges and smooth regions respectively. The values of the connection weights in each sub-network can be different, and we can even adopt different architectures and learning strategies for each sub-network for optimal processing of its assigned feature type.

Fuzzy Logic

From the previous description of fuzzy techniques, it is obvious that its main application in adaptive image processing will be to address the requirement of characterization , i.e., the specification of human visual preferences in terms of gray level value configurations. Many concepts associated with image processing are inherently fuzzy, such as the description of a region as "dark" or "bright", and the incorporation of fuzzy set theory is usually required for satisfactory processing results [25, 28, 45, 62, 105] . The very use of the words "textures", "edges" and "smooth regions" to characterize the basic image feature types implies fuzziness: the difference between smooth regions and weak textures can be subtle, and the boundary between textures and edges is sometimes blurred if the textural patterns are strongly correlated in a certain direction so that we can regard the pattern as multiple edges. Since the image processing system only recognizes gray level configurations, it will be natural to define fuzzy sets with qualifying terms like "texture", "edge" and "smooth regions" over the set of corresponding gray-level configurations according to

human preferences. However, one of the problems with this approach is that there is usually an extremely large number of possible gray level configurations corresponding to each feature type, and human beings cannot usually relate what they perceive as a certain feature type to a particular configuration. It is therefore one of the main contributions of this thesis that a *scalar* measure has been established which characterizes the degree of resemblance of a gray level configuration to either textures or edges. In addition, we can establish the exact interval of values of this measure where the configuration will resemble more to textures than edges, and *vice versa*. As a result, we can readily define fuzzy sets over this one-dimensional *universe of discourse* [61].

In addition, fuzzy set theory also plays an important role in the derivation of improved segmentation algorithms. A notable example is the *fuzzy c-means algorithm* [15, 40, 82, 94] , which is a generalization of the *k-means algorithm* [72] for data clustering. In the k-means algorithm, each data vector, which may contain feature values or gray level values as individual components in image processing applications, is assumed to belong to one and only one class. This may result in inadequate characterization of certain data vectors which possess properties common to more than one class, but then get arbitrarily assigned to one of those classes. This is prevented in the fuzzy c-means algorithm, where each data vector is assumed to belong to every class to a different degree which is expressed by a numerical membership value in the interval $[0, 1]$. This paradigm can now accommodate those data vectors which possess attributes common to more than one class, in the form of large membership values in several of these classes.

Evolutionary Computation

The often stated advantages of evolutionary computation include its implicit parallelism which allows simultaneous exploration of different regions of the search space [35] , and its ability to avoid local minima [7, 29] . However, in this thesis, we will emphasize its capability to search for the optimizer of *non-differentiable* cost function efficiently, i.e., to satisfy the requirement of optimization . An example of a non-differentiable cost function

in image processing would be the metric which compares the probability density function (pdf) of a certain local attribute of the image (gray level values, gradient magnitudes ,etc) with a desired pdf. We would, in general, like to adjust the parameters of the adaptive image processing system in such a way that the distance between the pdf of the processed image is as close as possible to the desired pdf. In other words, we would like to minimize the distance as a function of the system parameters. In practice, we have to approximate the pdf's using histograms of the corresponding attributes, which involves the counting of discrete quantities. As a result, although the pdf of the processed image is a function of the system parameters, it is not differentiable with respect to these parameters. Although stochastic algorithms like simulated annealing can also be applied to minimize non-differentiable cost functions, evolutionary computational algorithms represent a more efficient optimization approach due to the implicit parallelism of its population-based search strategy.

The relationship between the main classes of algorithms in computational intelligence and the major requirements in adaptive image processing is summarized in Figure 1.5.

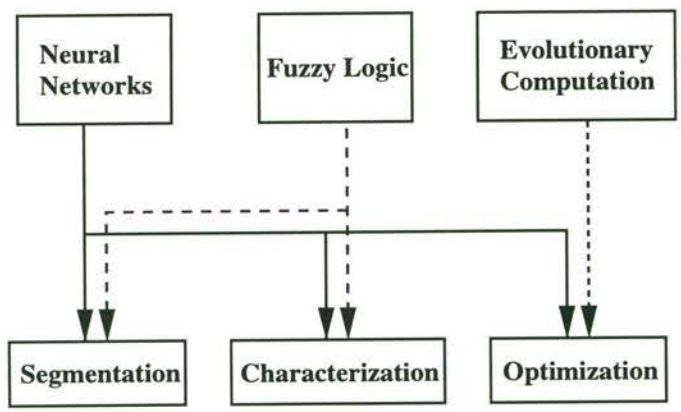


Figure 1.5: Relationships between the computational intelligence algorithms and the main requirements in adaptive image processing

1.2 Scope of the Thesis

In this thesis, as specific examples of adaptive image processing systems, we consider the *adaptive regularization* problem in image restoration [56, 68, 110] and the edge characterization problem in image analysis. We adopt the technique of neural network as a first approach to these problems due to its capability to satisfy all the three requirements in adaptive image processing, as illustrated in Figure 1.5. In particular, we use a specialized form of network known as *model-based neural network with hierarchical architecture* [21, 65]. The reason for its adoption is that its specific architecture, which consists of a number of model-based *sub-networks*, particularly facilitates the implementation of adaptive image processing applications, where each sub-network can be specialized to process a particular type of image features.

1.2.1 Adaptive Regularization

In regularized image restoration, the associated cost function consists of two terms: a data conformance term which is a function of the degraded image pixel values and the degradation mechanism, and the model conformance term which is usually specified as a continuity constraint on neighboring gray level values to alleviate the problem of ill-conditioning characteristic of this kind of inverse problems. The *regularization parameter* [47, 55] controls the relative contributions of the two terms toward the overall cost function.

In general, if the regularization parameter is increased, the model conformance term is emphasized at the expense of the data conformance term, and the restored image becomes smoother while the edges and textured regions become blurred. On the contrary, if we decrease the parameter, the fidelity of the restored image is increased at the expense of decreased noise smoothing. If a single parameter value is used for the whole image, it should be chosen such that the quality of the resulting restored image would be a compromise between the above two extremes.

More generally, we can adopt different regularization parameter values for regions in

the image corresponding to different feature types. This is more desirable due to the different noise masking capabilities of distinct feature types: since noise is more visible in the smooth regions, we should adopt a larger parameter value in those regions, while we could use a smaller value in the edge and textured regions to enhance the details there due to their greater noise masking capabilities. We can even further distinguish between the edge and textured regions and assign a still smaller parameter value to the textured regions due to their closer resemblance to noises.

Adaptive regularization can thus be regarded as a representative example of the design of an adaptive image processing system, since the stages of segmentation, characterization and optimization are included in its implementation: the segmentation stage consists of the partitioning of the image into its constituent feature types, the characterization stage involves specifying the desired gray level configurations for each feature type after restoration, and relating these configurations to particular values of the regularization parameter in terms of various image models and the associated cost functions. The final optimization stage searches for the optimal parameter values by minimizing the resulting cost functions. Since the hierarchical model-based neural networks can satisfy each of the above three requirements to a certain extent, we propose using such networks to solve this problem as a first step.

The selection of this particular image processing problem is by no means restrictive, as the current framework can be generalized to a large variety of related processing problems : in adaptive image enhancement [78, 117] , it is also desirable to adopt different enhancement criteria for different feature types. In adaptive image filtering, we can derive different sets of filter coefficients for the convolution mask in such a way that the image details in the edge and textured regions are preserved, while the noise in the smooth regions are attenuated. In segmentation-based image compression [91, 99, 106] , the partitioning of the image into its constituent features are also required in order to assign different statistical models and their associated optimal quantizers to the respective features.

1.2.2 Edge Characterization and Detection

The characterization of important features in an image requires the detailed specification of those pixel configurations which human beings would regard as significant. In this work, we consider the problem of representing human preferences, especially with regard to image interpretation, again in the form of a model-based neural network with hierarchical architecture [21, 65]. Since it is difficult to represent all aspects of human preferences in interpreting images using traditional mathematical models, we encode these preferences through a direct *learning* process, using image pixel configurations which humans usually regard as visually significant as training examples. As a first step, we consider the problem of edge characterization in such a network. This representation problem is important since its successful solution would allow computer vision systems to simulate to a certain extent the decision process of human beings when interpreting images.

Whereas the network can be considered as a particular implementation of the stages of segmentation and characterization in the overall adaptive image processing scheme, it can also be regarded as a self-contained adaptive image processing system on its own: the network is designed such that it automatically partitions the edges in an image into different classes depending on the gray level values of the surrounding pixels of the edge, and apply different detection thresholds to each of the classes. This is in contrast to the usual approach where a single detection threshold is adopted across the whole image independent of the local context. More importantly, instead of providing *quantitative* values for the threshold as in the usual case, the users are asked to provide *qualitative* opinion on what they regard as edges by manually tracing their desired edges on an image. The gray level configurations around the trace are then used as training examples for the model-based neural network to acquire an internal model of the edges, which is another example of the design of an adaptive image processing system through the training process.

As seen above, we have proposed the use of hierarchical model-based neural network for the solution of both these problems as a first attempt. It was observed later

that, whereas the edge characterization problem can be satisfactorily represented by this framework, resulting in adequate characterization of those image edges which humans regard as significant, there are some inadequacies in using this framework exclusively for the solution of the adaptive regularization problem, especially in those cases where the images are more severely degraded. These inadequacies motivate our later adoption of fuzzy set theory and evolutionary computation techniques, in addition to the previous neural network techniques, for this problem.

1.3 Contributions of the Current Work

With regard to the problems posed by the requirements of segmentation, characterization and optimization in the design of an adaptive image processing system, we have devised a system of interrelated solutions comprising the use of the main algorithm classes of computational intelligence techniques. The contributions of the work described in this thesis can be summarized as follows:

1.3.1 Application of Model-Based Neural Networks to Adaptive Regularization

A model-based neural network with hierarchical architecture [21, 65] is derived for the problem of adaptive regularization in image restoration. The image is segmented into smooth regions and combined edge/textured regions, and we assign a single sub-network to each of these regions for the estimation of the regional parameters. An important new concept arising from this work is our alternative viewpoint of the regularization parameters as model-based neuronal weights, which are then trainable through the supply of proper training examples. We derive the training examples through the application of *adaptive non-linear filtering* [87] to individual pixel neighborhoods in the image for an *independent* estimate of the current pixel value. Comparing with previous works where the regularization parameters are estimated using only information from the associated

iterative restoration process itself, or by using heuristic approaches , we believe this is the first time where the theory of adaptive non-linear filtering, typical examples of which include median filtering and weighted order-statistic (WOS) filtering [87] , are combined with the theory of iterative image restoration to provide independent information for the estimation of the regularization parameters.

1.3.2 Application of Model-Based Neural Networks to Edge Characterization

A model-based neural network with hierarchical architecture is proposed for the problem of edge characterization and detection. Unlike previous edge detection algorithms where various threshold parameters have to be specified [36, 50], this parameterization task can be performed implicitly in a neural network by supplying training examples. The most important concept in this part of the work is to allow human users to communicate their preferences to the adaptive image processing system through the provision of qualitative training examples in the form of edge tracings on an image, , which is a more natural way of specifying preferences for humans, than the selection of *quantitative* values for a set of parameters. With the adoption of this network architecture and the associated training algorithm, it will be shown that the network can generalize from sparse examples of edges provided by human users to detect all significant edges in images not in the training set. More importantly, no re-training and alteration of architecture is required for applying the same network to noisy images, unlike conventional edge detectors which usually require threshold re-adjustment.

As mentioned in the previous section, it was observed that the hierarchical model-based neural network can already model those human-preferred edge configurations in a satisfactory way, but that there are still some inadequacies in solving the adaptive regularization problem using this framework exclusively, which is primarily due to the different noise masking capabilities of the edges and textures in an image which requires

separate regularization strategies for each of them. This aspect of the regularization problem, which is not explicitly modelled in our previous network, in turn requires the determination of which pixel configurations correspond to edges and textures respectively. This motivates our development of the Edge-Texture Characterization (ETC) Measure, and its incorporation into a generalized hierarchical model-based network architecture which models the edges and textures as fuzzy sets. In addition to these neural network approaches, we also propose an alternative solution of the adaptive regularization problem which requires the use of evolutionary computational techniques. These generalizations are described below:

1.3.3 Formulation of the Edge-Texture Characterization (ETC) Measure

With regard to the problem of characterizing the degree of resemblance of a particular gray level configuration to either textures or edges, a novel measure ,known as the Edge-Texture Characterization (ETC) Measure, has been derived which quantifies this degree of resemblance through a *scalar* value. In addition, we have established exact intervals of this measure within which the corresponding gray level configuration resembles more to edges than textures, and *vice versa*. As a result, we can distinguish between edges and textures just by observing the value of this scalar quantity. This is particularly significant as we cannot usually distinguish between edges and textures by comparing the local variance or gradient magnitudes of the respective pixel configurations, which may contain similar levels of high frequency components. The current approach also represents an alternative form of segmentation compared with the usual approaches of distinguishing between different types of textures, and distinguishing between smooth regions and combined edge/textured regions.

1.3.4 Incorporation of the ETC Measure in a Fuzzy Model-Based NN for Adaptive Regularization

We have extended the previous framework of adaptive regularization using model-based neural network by incorporating the ETC measure to assign different regularization parameter values to edge and textured regions. This is necessary due to the different noise masking capabilities of the above two feature types. We have applied fuzzy techniques [61, 113] in this part of the work, where we define two fuzzy sets with the names EDGE and TEXTURE over the scalar ETC measure domain. This produces a fuzzy version of the adaptive regularization model-based network where we have included a so-called edge neuron and texture neuron in each sub-network, and which in turn estimate two different values of regularization parameters for each segmented image region. The final gray level estimate of each pixel is produced by combining the output of the two neurons in an augmented output layer, with the fuzzy membership values associated with the above two fuzzy sets as output weights.

1.3.5 Application of Evolutionary Programming to Adaptive Regularization

Apart from the neural network-based techniques, we have developed an alternative solution to the problem of adaptive regularization using evolutionary programming, which is a member of the class of evolutionary computational algorithms [7, 29] . Returning again to the ETC measure, we have observed that the distribution of the values of this quantity assumes a typical form for a large class of images. In other words, the shape of the probability density function (pdf) of this measure is similar across a broad class of images and can be modelled using piecewise continuous functions. On the other hand, this pdf will be different for blurred images or incorrectly regularized images. As a result, the model pdf of the ETC measure serves as a kind of *signature* for correctly regularized images, and we should minimize the difference between the corresponding pdf of the image being restored and the model pdf using some kind of distance measure. The requirement

to approximate this pdf using a histogram, which involves the counting of discrete quantities, and the resulting non-differentiability of the distance measure with respect to the various regularization parameters, necessitates the use of evolutionary computational algorithms for optimization . We have adopted evolutionary programming which, unlike genetic algorithm which is another widely applied member of this class of algorithms, operate directly on real-valued vectors instead of binary-coded strings and therefore more suited to the adaptation of the regularization parameters. In this algorithm, we have derived a parametric representation which expresses the regularization parameter value as a function of the local image variance. Generating a population of these *regularization strategies* which are vectors of the above hyperparameters, we apply the processes of mutation, competition and selection to the members of the population to obtain the optimal regularization strategy.

1.4 Organization of the Thesis

This thesis is divided into 8 chapters. The contents of each chapter are summarized as follows:

Chapter 2 reviews the neural network techniques and its application to image processing. Model-based neural networks are defined and the operations of two types of model-based neurons which will be used in subsequent chapters are described. We then generalize this framework to adopt a modular network architecture which incorporates clusters of model-based neurons as sub-networks.

Chapter 3 describes our work on applying model-based neural network with hierarchical architecture to the problem of adaptive regularization in image restoration. Experimental results are presented which illustrate the advantages and shortcomings of this scheme. It is the latter which motivates the development of the fuzzy version of the current algorithm in Chapter 6.

Chapter 4 describes our work on applying model-based neural network with hierarchical architecture to the problem of edge characterization and detection. We have presented

experimental results which compare the performances of this new approach with conventional edge detectors.

Chapter 5 describes the derivation of the Edge Texture Characterization (ETC) measure based on the representation of a local image pixel neighborhood as a correlated stochastic process. Expermental results are presented which illustrate the capability of this measure to distinguish between edges and textures in an image.

Chapter 6 begins by reviewing the fuzzy set techniques and its applications to image processing. We then describe a fuzzy model incorporating two fuzzy sets, EDGE and TEXTURE, which are specially defined to characterize their corresponding feature types in terms of ETC measure values. This model is incorporated into the previous model-based neural network to produce a fuzzified version of the network which is capable of assigning different regularization parameter values to edges and textures, aside from discriminating between smooth and abruptly varying image areas. Experimental results illustrate the advantages of adopting fuzzy techniques in comparison with the previous non-fuzzified network.

Chapter 7 describes the application of evolutionary programming to the adaptive regularization problem in image restoration. The properties of the probability density function (pdf) of the ETC measure are discussed and formalized in terms of a model ETC-pdf consisting of multiple piecewise continuous functions. We then define a distance measure between these pdf's. The resulting non-differentiability of the distance measure with respect to the regularization parameters necessitates the use of evolutionary programming for its optimization. Experimental results are presented which illustrate the performance of the EP-based algorithm with respect to conventional restoration algorithms, and the previous neural network-based algorithms.

Chapter 8 presents the conclusion of this thesis and provides suggestions for future research directions.

Chapter 2

Model-Based Neural Network with Hierarchical Architecture (HMBNN)

2.1 Introduction

In this Chapter, we introduce the technique of neural networks, which is the first computational intelligence algorithm to be applied to our problem of adaptive regularization in image restoration.

Neural network [2, 41, 42] represents an alternative computational paradigm to the traditional serial (or Von Neumann type) machines. Whereas serial computers excel in specific computational tasks which involve large amounts of numerical calculations, its performance in other tasks, such as visual perception and pattern recognition, is not directly comparable with the capabilities of human beings. This performance gap is usually attributed to the fundamental difference in architecture of Von Neumann computers and the human brain. In contrast with the execution of an algorithm on a serial machine, where the operations of the algorithm are codified in the form of *program instructions* and executed one after another in a sequential manner through a single central processing unit, the cerebral cortex consists of a vast number of comparatively simple information processors in the form of neurons. In addition, the neurons are densely interconnected with each other through their *axons* and *dendrites* [2, 52, 102] . In this way, massively

parallel processing of information is facilitated where any processing "algorithms", if such words are permitted, are encoded in a distributed manner as the simultaneous firing patterns of specific neuron subsets within the cerebral cortex. The resulting signals are then propagated through the densely interconnected network to other neurons. Usually, each particular neuron will be receiving information from a large number of neurons through its dendrites, and will in turn broadcast information through its axon to a large number of other neurons.

This architecture partly explains the remarkable capability of human beings in performing visual pattern recognition: this distributed encoding enables different neurons to receive and process information from different parts of a visual scene *simultaneously.* The processed information from individual neurons is then combined through higher level neurons to reconstruct a consistent global interpretation of the entire scene. This is unlike the operation of a serial machine where only a restricted window of information is available from the scene for processing in a specific time interval.

More importantly, the overall behavior of the biological neural network can be modified through the process of *learning.* This is usually performed by modifying the properties of *synapses* [2, 52, 102] , which serve as gateways coupling the body of a neuron with axons or dendrites, in response to external stimulations. The resulting flexibility gained through this adaptive process allows the network to quickly acquire the essential characteristics of novel environment. In addition, the large number of neurons and the distributed encoding lend a degree of fault tolerance to the network, such that its performance deteriorates gracefully with the malfunction of individual neurons rather than abruptly.

Artificial neural network represents the first attempts to incorporate the above desirable properties into computing machines. Corresponding to the neurons in the biological network, we define artificial neurons which perform simple mathematical operations. These artificial neurons are connected with each other through *network weights* which specify the strength of the connection. Analogous to its biological counterpart, these network weights are adjustable through a learning process which enables the network to perform a variety of computational tasks. The neurons are usually arranged in *layers,*

with the input layer accepting signals from the external environment, and the output layer emitting the result of the computations. In between these two layers are usually a number of *hidden* layers which perform the intermediate steps of computations. The architecture of a typical artificial neural network with one hidden layer is shown in Figure 2.1. In specific types of network, the hidden layers may be missing and only the input and output layers are present.

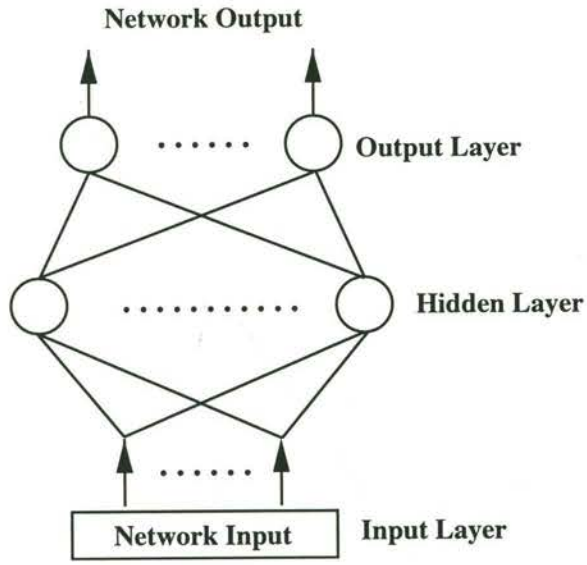


Figure 2.1: The architecture of a neural network with one hidden layer

The learning algorithms for the artificial neural networks are usually divided into two main classes, the supervised learning algorithms and unsupervised learning algorithms, which are described below.

2.1.1 Supervised Learning Algorithms

For this class of learning algorithms, the individual *training example* consists of the pair $(\mathbf{x}, \mathbf{d})$, where $\mathbf{x}$ is the *input vector* and $\mathbf{d}$ is the *desired output vector*. The connection weights are adjusted in such a way that , given the input vector $\mathbf{x}$, the output of the network will approximate the desired output vector $\mathbf{d}$. This class of algorithms is especially useful in learning classification problems when the ground truth is available for a subset of data, and in function interpolation problems given a set of sample points from

the original function.

2.1.2 Unsupervised Learning Algorithms

For this class of algorithms, the individual training example consists of the input vector $\mathbf{x}$ only. The weight adjustment algorithm is designed in such a way that the network gradually develops an internal representation of the intrinsic structure of the data. This class of algorithms is especially useful for data mining applications where the objective is to discover some hidden structures within the set of data. It is also useful for optimally summarizing a set of high-dimensional data vectors in terms of their projections onto a specific data subspace, a notable example being the extraction of features using principal component analysis (PCA) [11, 76, 77] .

We will apply both algorithm classes to our adaptive image processing problem later in the thesis: in Chapter 3, we reformulate the problem of adaptive regularization in terms of the training of a *model-based* neural network with hierarchical architecture [65], where we adapt the weights of the network using a supervised learning strategy . In Chapter 4, we apply competitive learning [60], which is an unsupervised learning strategy, to capture the essential characteristics of image visual features which human beings regard as significant.

2.2 Model-Based Neural Network

Instead of adopting the general architecture in Figure 2.1 for our adaptive image processing applications, we propose the use of *modular model-based* neural network for our purpose. We used the term "model-based" in the sense of Caelli *et.al* [21] , where expert knowledge in the problem domain is explicitly incorporated into a neural network by restricting the domain of the network weights to a suitable subspace in which the solution of the problem resides. In this way, a single weight vector can be uniquely specified by a small number of parameters and the "curse of dimensionality" problem is partially alleviated.

To appreciate this formulation more readily, we review some fundamental concepts of

artificial neuron computation, where each such neuron is the elementary unit of computation in a neural network [41, 42] . In general, the s-th neuron in the network implements a mapping $f_s : \mathbf{R}^N \longrightarrow \mathbf{R}$ which is given by

$$\begin{aligned} y_s &= f_s(\mathbf{x}) \\ &= g(\mathbf{p}_s^T \mathbf{x}) \\ &= g(\sum_{n=1}^{N} p_{qn} x_n) \end{aligned} \tag{2.1}$$

where

$$\begin{aligned} \mathbf{x} &= [x_1, \ldots, x_N]^T \in \mathbf{R}^N \quad \text{and} \\ \mathbf{p}_s &= [p_{s1}, \ldots, p_{sN}]^T \in \mathbf{R}^N \end{aligned}$$

are the input vector and the weight vector for the neuron respectively. g is usually a nonlinear sigmoid function which limits the output dynamic range of the neuron. We will extend this concept and define two types of model-based neuron in the next two sections.

2.2.1 Weight-parameterized model-based neuron

The main assumption in this weight-parameterized model-based formulation [21] is that for a specific domain of knowledge, the corresponding weight domain is restricted to a low-dimensional submanifold of $\mathbf{R}^N$.

Denoting this weight domain by $\mathcal{W}_p$, the formulation thus assumes the existence of a mapping $\mathcal{M} : \mathbf{R}^M \longrightarrow \mathcal{W}_p \subset \mathbf{R}^N$ such that

$$\mathbf{p} = \mathcal{M}(\mathbf{z}) \tag{2.2}$$

where

$$\begin{aligned} \mathbf{z} &= [z_1, \ldots, z_M]^T \in \mathbf{R}^M \\ \mathbf{p} &= [p_1, \ldots, p_N]^T \in \mathbf{R}^N \\ &= [\mathcal{M}_1(\mathbf{z}), \ldots, \mathcal{M}_N(\mathbf{z})]^T \end{aligned}$$

with $M < N$. The mappings $\mathcal{M}_n : \mathbf{R}^M \longrightarrow \mathbf{R}, n = 1, \ldots, N$ are the component functions of $\mathcal{M}$. The structure of a typical weight-parameterized model-based neuron is shown in Figure 2.2.

Assuming that each component function is differentiable with respect to $\mathbf{z}$, the steepest descent update rule for the components of $\mathbf{z}$ is as follows

$$\begin{aligned} z_m(t+1) &= z_m(t) - \eta \frac{\partial \mathcal{E}_t}{\partial z_m} \\ &= z_m(t) - \eta \sum_{n=1}^{N} \frac{\partial \mathcal{E}_t}{\partial p_n} \frac{\partial p_n}{\partial z_m} \end{aligned} \tag{2.3}$$

where $\mathcal{E}_t$ is an instantaneous error measure between the network output and the desired output.

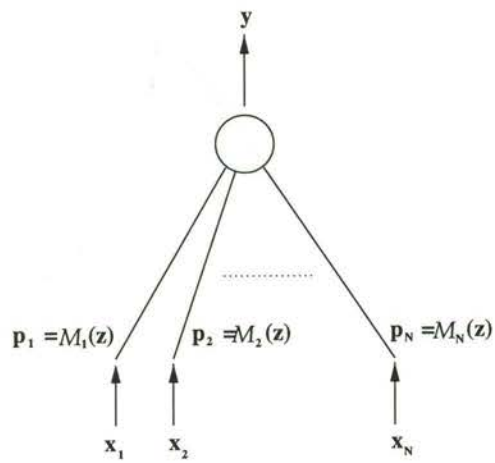


Figure 2.2: The weight-parameterized model-based neuron

As a result, if we possess prior knowledge of our problem domain in the form of the mapping $\mathcal{M}$ and if $M \ll N$, the optimization can proceed within a subspace of greatly reduced dimensionality and the problem of "curse of dimensionality" is partially alleviated. In the next Chapter we will present an alternative formulation of the adaptive regularization problem in terms of this model-based neuron.

2.2.2 Input-parameterized model-based neuron

Instead of mapping the embedded low-dimensional weight vector $\mathbf{z} \in \mathbf{R}^M$ to the high-dimensional weight vector $\mathbf{p} \in \mathbf{R}^N$, an alternative implementation of a model-based

neuron, which we will call the *input-parameterized* model-based neuron, is to map the high-dimensional input vector $\mathbf{x} \in \mathbf{R}^N$ to a low-dimensional vector $\mathbf{x}^P \in \mathbf{R}^M$. This is under the assumption that for the problem at hand, the reduced vector $\mathbf{x}^P$ can fully represent the essential characteristics of its higher dimensional counterpart $\mathbf{x}$. If such a mapping exists for the set of input vectors $\mathbf{x}$, we can directly represent the weight vector in its low-dimensional form $\mathbf{z}$ in the network instead of its embedded form in a high-dimensional vector $\mathbf{p}$.

More formally, we assume the existence of a mapping $\mathcal{P} : \mathbf{R}^N \longrightarrow \mathbf{R}^M$, such that $\mathbf{x}^P = \mathcal{P}(\mathbf{x}) \in \mathbf{R}^M$, where $M < N$. The operation of this model-based neuron is then defined as follows:

$$\begin{aligned} y_s &= f_s(\mathbf{x}) \\ &= g(\mathbf{z}_s^T \mathbf{x}^P) \\ &= g(\sum_{m=1}^{M} z_{sm} x_m^P) \end{aligned} \tag{2.4}$$

The weights of the network can then be adjusted directly using the following equation

$$z_m(t+1) = z_m(t) - \eta \frac{\partial \mathcal{E}_t}{\partial z_m} \tag{2.5}$$

without involving the high-dimensional weight vector $\mathbf{p}$. The input-parameterized model-based neuron is illustrated in Figure 2.3.

Both types of model-based neurons allow the search for the optimal weights to proceed more efficiently in a low-dimensional weight space. The decision to use either one of the above neurons depend on the nature of the problem, which in turn determine the availability of either the mapping $\mathcal{M}$ for the weights, or the mapping $\mathcal{P}$ for the input: if the parameter space of a certain problem is restricted to a low-dimensional surface in a high-dimensional space, then it is natural to adopt the weight-parameterized model-based neuron. This is the case for the adaptive regularization problem in Chapter 3, where the valid set of weight vectors for the network forms a path, or alternatively a one-dimensional surface, governed by the single regularization parameter λ.

On the other hand, for cases where such representation for the weight vector is not readily available, while there is evidence , possibly through the application of principal

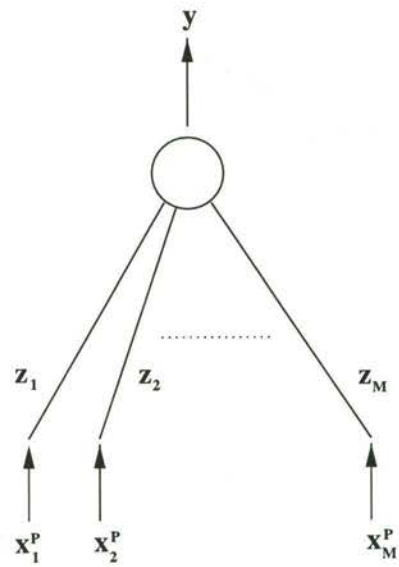


Figure 2.3: The input-parameterized model-based neuron

component analysis (PCA) [11, 76, 77] or prior knowledge regarding the input vector distribution, of the existence of an information-preserving operator (with respect to the current problem) which maps the input vectors to a low-dimensional subspace, then it is more natural to adopt the input-parameterized model-based neuron. This is the case for the edge characterization model-based neural network in Chapter 4, where the original input vector comprising a window of edge pixel values is mapped to a two-dimensional vector representing the two dominant gray levels around the edge.

2.3 Hierarchical Neural Network Architecture

While the previous model-based formulation describes the operations of individual neurons, the term "modular neural network" [1, 38, 48, 58, 79, 108, 107] refers to the overall architecture where specific computations are localized within certain structures known as *sub-networks*. This class of networks serves as natural representations of training data arising from several distinct classes, where each such class is represented by a single sub-network. Specific implementations of modular neural networks include the various hierarchical network architecture described in [31, 64, 65, 69, 70].

We are especially interested in the particular hierarchical implementation of the modu-

lar network structure by Kung and Taur [65]. Their proposed architecture associates a *sub-network output* $\phi(\mathbf{x}, \mathbf{w}_r)$ with the r-th sub-network. In addition, we define lower-level *neurons* within each sub-network with their corresponding *local neuron output* $\psi_r(\mathbf{x}, \mathbf{w}_{s_r})$, $s_r = 1, \ldots, S_r$. Depending on the specific relationships between the sub-network output $\phi(\mathbf{x}, \mathbf{w}_r)$ and its associated local neuron output $\psi_r(\mathbf{x}, \mathbf{w}_{s_r})$, Kung and Taur has defined two main categories of hierarchical architecture:

2.3.1 Hidden-Node Hierarchical Architecture

For this class of hierarchical architecture, the sub-network output is defined as the linear combination of the local neuron outputs as follows:

$$\phi(\mathbf{x}, \mathbf{w}_r) = \sum_{s_r=1}^{S_r} c_{s_r} \psi_r(\mathbf{x}, \mathbf{w}_{s_r}) \tag{2.6}$$

where c_{s_r} are the combination coefficients. The local neuron output evaluation can assume a variety of forms depending on the current application. The most important functions include the sigmoid function:

$$\psi_r(\mathbf{x}, \mathbf{w}_{s_r}) = g(\mathbf{w}_{s_r}^T \mathbf{x}) \tag{2.7}$$

where $g(u)$ is the sigmoid function

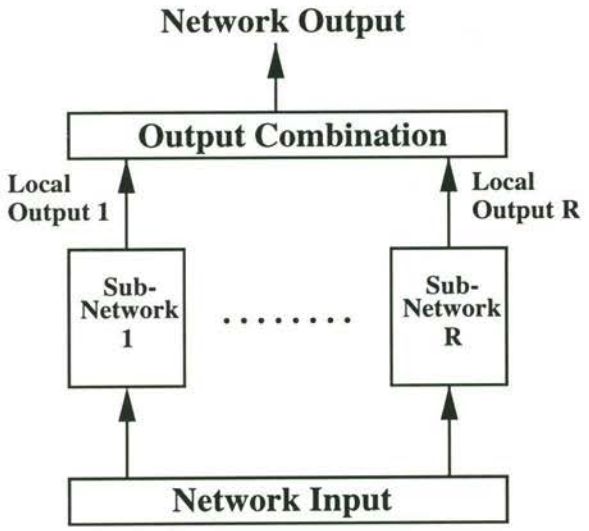
$$g(u) = \frac{1}{1 + e^{-u}} \tag{2.8}$$

and the Gaussian radial basis function (RBF)

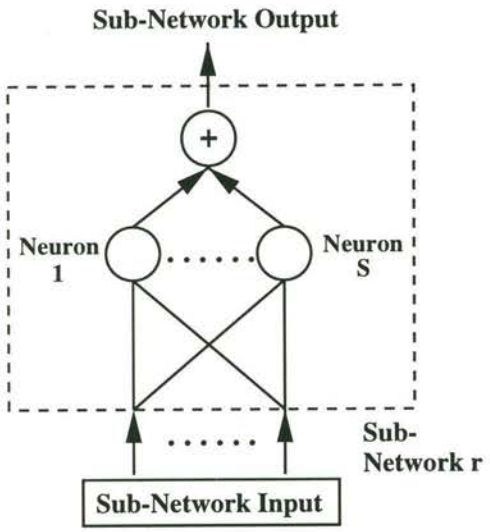
$$\psi_r(\mathbf{x}, \mathbf{w}_{s_r}) = e^{-\frac{\|\mathbf{x} - \tilde{\mathbf{w}}_{s_r}\|^2}{2\sigma_{s_r}^2}} \tag{2.9}$$

where $\mathbf{w}_{s_r} = [\tilde{\mathbf{w}}_{s_r}^T \quad \sigma_{s_r}^2]^T$.

Due to the resemblance of each sub-network to a two-layer feedforward neural network, the current architecture is especially suitable for function approximation applications, where a complex function is divided into several comparatively simple components and with each of these assigned to a single sub-network. A supervised training algorithm is then applied to each sub-network for weight adjustment with a corresponding partition of the training examples. The hidden-node hierarchical architecture and the structure of its sub-network is shown in Figure 2.4.



(a)



(b)

Figure 2.4: Hidden-node hierarchical network architecture (a) global network architecture (b) sub-network architecture

2.3.2 Subcluster Hierarchical Architecture

For this class of networks, a winner-take-all competition process is applied to all the neurons within a certain sub-network to determine the *local winner* as follows

$$\mathbf{w}_{s_r^*} = \arg\min_{s_r} \psi_r(\mathbf{x}, \mathbf{w}_{s_r}) \tag{2.10}$$

where $\mathbf{w}_{s_r^*}$ is the index of the winning neuron.

The sub-network output $\phi(\mathbf{x}, \mathbf{w}_r)$ is then substituted with the corresponding local neuron output of the winner

$$\phi(\mathbf{x}, \mathbf{w}_r) = \psi_r(\mathbf{x}, \mathbf{w}_{s_r^*}) \tag{2.11}$$

In accordance with this competition process, it is natural to adopt the Euclidean distance for evaluating the local neuron output

$$\psi_r(\mathbf{x}, \mathbf{w}_{s_r}) = \|\mathbf{x} - \mathbf{w}_{s_r}\| \tag{2.12}$$

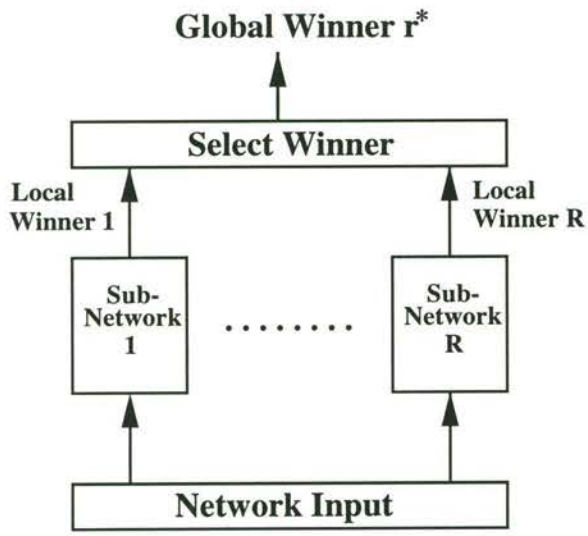
This class of networks is especially suitable for unsupervised pattern classification, where each pattern class is composed of several disjoint subsets of slightly different characteristics. We can then assign each primary pattern class to a single sub-network, and each secondary class under the current primary class to a neuron within the sub-network. The subcluster hierarchical network architecture and the structure of its sub-network is shown in Figure 2.5.

2.3.3 Combination of Sub-Network Outputs

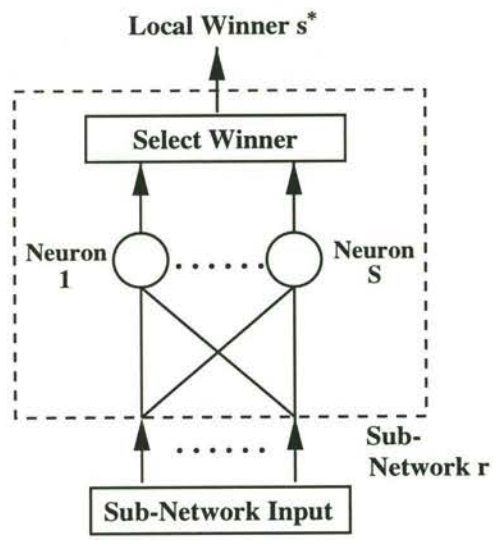
In the last stage, the information provided by the various sub-networks is combined to form the final network output. In the context of classification problems, Kung and Taur defined the final network output as the index of the *global winner* using a winner-take-all approach

$$r^* = \arg\max_{r} \phi(\mathbf{x}, \mathbf{w}_r) \tag{2.13}$$

This definition is applicable to both the hidden-node hierarchical architecture and the subcluster hierarchical architecture.



(a)



(b)

Figure 2.5: Subcluster hierarchical network architecture (a) global network architecture (b) sub-network architecture

Whereas the index r^*, which indicates the assigned class of the current network input, is important for classification applications, sometimes the value of the network output itself is important, as in function approximation problems. In the latter case, we usually define the network output y as a function of the individual sub-network output as follows:

$$y = f(\phi(\mathbf{x}, \mathbf{w}_1), \ldots, \phi(\mathbf{x}, \mathbf{w}_R)) \tag{2.14}$$

where R is the number of sub-networks.

A common example for f is the linear combination operation, where

$$y = \sum_{r=1}^{R} a_r \phi(\mathbf{x}, \mathbf{w}_r) \tag{2.15}$$

and a_r are the combination coefficients. However, the form for f is in general not restricted to linear operations, and we can admit alternative, possibly nonlinear, form for f depending on the particular problem at hand.

2.4 Model-Based Neural Network with Hierarchical Architecture (HMBNN)

Given the previous description of the model-based neuron, which specifies the computational operations at the neuronal level, and the overall hierarchical structure, which specifies the macroscopic network architecture, it is natural to combine the two in a single framework. More specifically, we can incorporate model-based neuron computation at the lowest hierarchy, i.e., at the level of single neuron within a sub-network, of the hierarchical neural network.

Formally, we can specify the computation of the local neuron outputs $\psi_r(\mathbf{x}, \mathbf{w}_{s_r})$ of the neurons in two different ways, in accordance with the previous two definitions of model-based neuron computation. For the weight-parameterized model-based neuron, and assuming $\mathbf{w}_{s_r} \in \mathbf{R}^N$, the local neuron output computation is specified as follows:

$$\psi_r(\mathbf{x}, \mathbf{w}_{s_r}) \equiv \psi_r(\mathbf{x}, \mathcal{M}(\mathbf{z}_{s_r})) \tag{2.16}$$

where $\mathbf{z}_{s_r} \in \mathbf{R}^M$, with $M < N$, is the model-based weight vector, and $\mathcal{M} : \mathbf{R}^M \longrightarrow \mathbf{R}^N$ is the mapping relating the model-based weight vector to the neuronal weight vector. In this case, the lower-dimensional model-based vector $\mathbf{z}_{s_r}$ is embedded in the higher-dimensional weight vector $\mathbf{w}_{s_r}$, and the weight optimization is carried out in the lower-dimensional weight space $\mathbf{R}^M$.

For the input-parameterized model-based neuron, instead of embedding the model-based vector $\mathbf{z}_{s_r}$ in a higher-dimensional space, we map the high-dimensional input vector $\mathbf{x} \in \mathbf{R}^N$ on to a low-dimensional submanifold $\mathbf{R}^M$ through the operator $\mathcal{P}$ as follows:

$$\psi_r(\mathbf{x}, \mathbf{w}_{s_r}) \equiv \psi_r(\mathbf{x}^P, \mathbf{z}_{s_r}) \tag{2.17}$$

$$= \psi_r(\mathcal{P}(\mathbf{x}), \mathbf{z}_{s_r}) \tag{2.18}$$

where $\mathbf{x}^P = \mathcal{P}(\mathbf{x})$ is the low-dimensional input vector corresponding to $\mathbf{x}$ in $\mathbf{R}^N$.

These two implementations of model-based neurons are equally applicable to both the hidden-node hierarchical architecture and subcluster hierarchical architecture, thus giving rise to four different classes of model-based neurons with hierarchical architecture. We have selected two among these four classes of networks for our adaptive image processing applications, which are described in the next section.

2.5 HMBNN for Adaptive Image Processing

2.5.1 Adaptive Regularization in Image Restoration

As mentioned in Chapter 1, our formulation of the adaptive regularization problem requires the partition of an image into disjoint regions, and the assignment of the optimal regularization parameter value λ to the respective regions. It is natural, therefore, to assign a single sub-network to each such region, and regard the regularization parameter λ as a model-based weight to be optimized using a special set of training examples. More specifically, for the r-th sub-network, it will be shown in the next Chapter that the image restoration process is characterized by the evaluation of the local neuron output as a linear

operation as follows.

$$\psi_r(\mathbf{x}, \mathbf{p}_{s_r}) = \mathbf{p}_{s_r}^T \mathbf{x} \tag{2.19}$$

where $\mathbf{x} \in \mathbf{R}^N$ denotes a vector of image gray level values in a local neighborhood, $\mathbf{p}_{s_r}$ is the *image restoration convolution mask* derived from the point spread function (PSF) of the degradation mechanism. The dimension N of the vectors $\mathbf{x}$ and $\mathbf{p}_{s_r}$ depends on the size of the PSF. It will be shown that the convolution mask coefficient vector $\mathbf{p}_{s_r}$ can be expressed as a function of the regularization parameter λ. In other words, there exists a mapping $\mathcal{M} : \mathbf{R} \longrightarrow \mathbf{R}^N$ such that

$$\mathbf{p}_{s_r} = \mathcal{M}(\lambda) \tag{2.20}$$

which is equivalent to the embedding of the scalar parameter λ in the high-dimensional space $\mathbf{R}^N$, and this neuron thus corresponds to a weight-parameterized model-based neuron.

The training examples for this network is of the form $(\mathbf{x}, \triangle x)$, where $\triangle x$, the desired output of the network, corresponds to the required change in gray level value for the current pixel to achieve satisfactory restoration. Due to the continuous range of $\triangle x$, the current problem belongs to the category of function approximation applications which favors the adoption of the hidden-node hierarchical architecture. We have therefore employed this architecture for the current adaptive regularization problem with the model-based weight λ in each sub-network determined by supervised learning.

2.5.2 Edge Characterization and Detection

In Chapter 4, we will introduce an adaptive edge characterization scheme, where different threshold parameters are defined for detecting edges under different background illuminations. This is motivated by our observation of the different preferences of human beings in regarding a certain magnitude of gray level discontinuity as constituting a significant edge feature under different illuminations. To incorporate this criterion into our edge detection process, it is natural to adopt a hierarchical network architecture where we designate

each sub-network to represent a different illumination level, and each neuron in the sub-network to represent different prototypes of edge-like features under the corresponding illumination level. In Chapter 4, we have defined an *edge prototype* as a two-dimensional vector $\mathbf{w} \in \mathbf{R}^2$ which represents the two dominant gray level values on both sides of the edge.

Due to this necessity of inferring prototypes from the human-supplied training examples, it is natural to adopt unsupervised competitive learning where each prototype is represented as the weight vector of a neuron. The winner-take-all nature of the competition process also favors the use of the subcluster hierarchical architecture, such that only the local winner within each sub-network is allowed to update its weight vector. In addition, for the r-th sub-network, the local neuron output is evaluated in terms of the Euclidean distance between the edge prototype and the current edge example

$$\psi_r(\mathbf{x}^P, \mathbf{z}_{s_r}) = \|\mathbf{x}^P - \mathbf{z}_{s_r}\| \tag{2.21}$$

where $\mathbf{x}^P \in \mathbf{R}^2$ is the current edge example, and $\mathbf{z}_{s_r}$ is the s_r-th edge prototype of the r-th sub-network.

In view of the fact that the current edge example is usually specified in terms of a window of gray level values $\mathbf{x} \in \mathbf{R}^N$, where $N \gg 2$, it is necessary to summarize this high-dimensional vector in terms of its two dominant gray level values. In other words, we should derive a mapping $\mathcal{P} : \mathbf{R}^N \longrightarrow \mathbf{R}^2$ such that

$$\mathbf{x}^P = \mathcal{P}(\mathbf{x}) \tag{2.22}$$

which corresponds to an input-parameterized model-based neuron.

Chapter 3

Application of HMBNN to Adaptive Regularization

3.1 Introduction

In this Chapter, we apply the model-based neural network with hierarchical architecture (HMBNN), defined in the previous Chapter, to the problem of adaptive regularization in image restoration. Image restoration techniques [3, 55] are important in that they can extract substantial information from only a degraded version of visual data originating from some physical events, and may even be indispensable if the event itself is non-repeatable. Degradations are usually in the form of blurring due to lens defocusing, atmospheric turbulence, and relative motion which are collectively known as system degradations, and noises arising at various points in the imaging system which are collectively known as statistical degradations [3, 55, 67].

Image restoration is in general a difficult problem due to the ill-conditioned or even ill-posed nature of the associated inverse filtering operation [3, 14, 67]. Regularization is one of the most popular techniques employed to alleviate this situation. In regularized image restoration, the cost function consists of a data-conformance evaluation term and a model-conformance evaluation term [14, 88]. The model-conformance term is usually specified as a continuity constraint on neighboring gray-level values in image processing applications.

The contribution of these two terms is adjusted by the so-called regularization parameter or the Lagrange multiplier which allows the two terms to combine additively:

$$E = \frac{1}{2}\|\mathbf{y} - \mathbf{H}\hat{\mathbf{x}}\|^2 + \frac{1}{2}\lambda\|\mathbf{D}\hat{\mathbf{x}}\|^2 \tag{3.1}$$

where the vector $\mathbf{y}$, with $y_i, i = 1, \ldots, N_I$ as components and N_I as the number of image pixels, denotes the blurred image with its pixel values lexicographically ordered. The vectors $\hat{\mathbf{x}}$ with components $\hat{x}_i$ is the corresponding restored image and the matrix $\mathbf{H}$, with components $h_{ij}, i, j = 1, \ldots, N_I$, is the blur function. The matrix $\mathbf{D}$ with components d_{ij} is a linear operator on $\hat{\mathbf{x}}$ and λ is the regularization parameter. For image restoration purpose, $\mathbf{D}$ is usually a differential operator, and the minimization of the above cost function effectively limits the local variations of the restored image. Iterative approaches are usually employed to search for the minimizer of this cost function. Interpreting Eq (3.1) in a probabilistic setting, it can be shown that the traditional Wiener filter for image restoration can be considered a special case within this general regularization framework [3].

Notwithstanding the improvement of the regularization approach upon the original inverse filtering approach, the restored image is still plagued by two primary types of distortions, namely, noises and ringing artifacts. The amplification of the former is brought under control to a certain extent by regularization. The latter is a result of the high frequency response mismatch of the regularized filter to the inverse filter , and is manifested as alternate dark and bright stripes near the major edges [67, 101]. The ringing artifacts can be suppressed by either decreasing the level of regularization in the edge region to reduce the filtering mismatch, or by increasing the regularization in the edge's vicinity to enforce gray-level continuity.

From the above discussion, it is obvious that an adaptive approach of regularization has to be adopted. Determining the correct value of the regularization parameter at each pixel site is a non-trivial problem, as the magnitude of the parameter has no obvious relationship with the desired regional image quality. In theory, an optimal parameter value exists for each pixel site if we view the total restoration error as being composed of two terms, namely the noise amplification error term and the regularization error

term [67, 68]. As the former is monotonically decreasing with the parameter value while the latter is monotonically increasing with it, there exists an optimal parameter value which minimizes the overall restoration error. Attempts have been made to estimate this optimum parameter iteratively [32, 53, 56, 57], but the value is only optimal with respect to the particular restoration error functional adopted, which in turn depends strongly on the particular regularization operator used. As a result, optimization of the parameter value with respect to a particular functional may not necessarily correspond to improved visual quality for the image.

Notwithstanding the above difficulties, we still possess the qualitative knowledge that small regularization parameters are to be used for the edge and textured regions to enhance the details there, while large parameters are required for the smooth regions to suppress the noise and ringing, although the exact parameter value for attaining optimum visual quality is in general unknown. This strongly suggests an adaptive approach in which the functional dependence between the parameter value and the local image activity is empirically determined, as exemplified by the works in [56, 68, 90]. The work in [85] can be considered a discrete version of the previous approaches where the continuum of parameter values is replaced by a finite number of values, with each of them corresponding to a particular range of image activities. As usual, the difficulties of these approaches lie in the non-intuitiveness of the regularization parameter value as a specification of the local image quality, and considerable experimentations are required to determine the correct function relating the regularization parameter value to the local image activity. Kang and Katsaggelos, in their spatially adaptive iterative restoration algorithm [56], have incorporated limited adaptivity in the determination of this functional form by automatically adjusting a multiplicative factor of this function based on particular global attributes of the partially restored image.

An example of a more intuitive image quality specification can be found in the field of nonlinear adaptive image filtering, where we express our desire to preserve certain features in different image regions by indirectly specifying a different filter mask for each such region [5, 87]. The distribution of the coefficient values in different masks directly

reflects the degree of detail preservation desired in different image regions. As a result, the above problem of regularization parameter assignment can be alleviated if we can first specify the desired local image quality in terms of a filter mask and then somehow define a correspondence between the mask and a particular parameter value.

In this Chapter, we propose an alternative solution to the problem of regularization parameter assignment by realizing the above correspondence in the form of an HMBNN [21, 65]. A model-based neuron is in effect a conventional artificial neuron with its weights mutually coupled through a small number of parameters, which serves to reduce the dimension of the underlying network weight space. We will show that the original image restoration problem can be interpreted as the computation of a model-based neuron with the regularization parameter being the coupling parameter between its various weights. The desired restoration behavior in a particular image region can then be specified in terms of a *regional training set* generated by the predicted pixel values of a pre-specified *regional filter mask*. We can then associate each distinct image region with a model-based neuron, and apply the training process to each neuron to vary its weights, or in effect the regional regularization parameter, to best approximate the local desired characteristics. As a result, the current assignment is more relevant in relation to the local spatial characteristics of the image than the usual practice of using an arbitrary function of the signal to noise ratio to determine the parameter value [56, 67].

3.2 The Hopfield Neural Network Model for Image Restoration

3.2.1 Optimization of the Restoration Cost Function

In the current work, the optimization of the primary image restoration cost function E was performed within the framework of neural network optimization where a modified neural network architecture based on the Hopfield neural network model in [44, 81, 118] was employed. This *primary* Hopfield image restoration neural network ,which is mainly

responsible for optimizing the value of E, is to be carefully distinguished from the HMBNN described previously, which serves as a *secondary* neural network to optimize the regularization parameter λ in E.

Image restoration employing the Hopfield neural network model [44] was first proposed by Zhou *et.al* [118] where the parameters characterizing the point spread function (PSF) were embedded into the connection weights of the network, and the energy-minimizing capability of the network was then exploited to minimize this cost function as it evolved into its equilibrium state.

The energy E_H of the Hopfield network is defined in terms of the connection weights between the neurons, the bias input of each neuron and the collective states of each individual neuron at a particular instant as follows.

$$E_H = -\frac{1}{2}\sum_{i=1}^{N_I}\sum_{j=1}^{N_I} w_{ij}v_i v_j - \sum_{i=1}^{N_I} b_i v_i \tag{3.2}$$

where v_i is the output of the i-th neuron, w_{ij} is the connection weight between the i-th and the j-th neuron, b_i is the bias input for the i-th neuron, and N_I is the number of neurons in the network.

Comparing the intrinsic neural network cost function E_H with the restoration cost function E, we notice that both equations are quadratic in the variables v_i if we identify the gray-level values of the restored image $\hat{x}_i$ with v_i and the number of neurons N_I with the number of image pixels. The connection weights and input bias of the neural network are pre-computed by equating the remaining corresponding variables in the two equations.

$$w_{ij} = -\sum_{p=1}^{N_I} h_{pi}h_{pj} - \lambda\sum_{p=1}^{N_I} d_{pi}d_{pj} \tag{3.3}$$

$$b_i = \sum_{p=1}^{N_I} y_p h_{pi} \tag{3.4}$$

where h_{pi} and d_{pi} are the elements of $\mathbf{H}$ and $\mathbf{D}$ respectively.

In the original implementation of the network by Zhou *et.al* [118], they followed the canonical update procedure for the Hopfield neurons: the variable u_i is defined for the i-th neuron as follows:

$$u_i = \sum_{j=1}^{N_I} w_{ij}v_j + b_i \tag{3.5}$$

The required amount of update $\triangle v_i$ for the i-th pixel was then evaluated by passing u_i through a hard-limiting nonlinearity $g(\cdot)$:

$$\triangle v_i = g(u_i) \tag{3.6}$$

where

$$g(x) = \begin{cases} 1 & \text{if } x > 0 \\ 0 & \text{if } x = 0 \\ -1 & \text{if } x < 0 \end{cases} \tag{3.7}$$

The updated gray level value for the i-th pixel is then given by

$$v_i(t+1) = v_i(t) + \triangle v_i \tag{3.8}$$

From these equations, it is seen that the gray level value $v_i(t+1)$ can change by at most 1 during each update, thus resulting in a long time for convergence. In view of this, Guan [37] has derived an equivalent update procedure which allows $\triangle v_i$ to be greater than 1 to increase the speed of convergence. In other words, the original Hopfield framework is relaxed in favor of more efficient operations. In this algorithm, the neurons in the network are updated sequentially by equating the derivative of E with respect to each v_i to zero and solving for the new output $v_i(t+1)$, thus in effect performing coordinate descent [71].

$$\frac{\partial E}{\partial v_i} = -(\sum_{j=1}^{N_I} w_{ij} v_j + b_i) = -u_i = 0 \qquad \forall i \tag{3.9}$$

where

$$u_i = \sum_{j=1}^{N_I} w_{ij} v_j + b_i \tag{3.10}$$

is the *activation* of the neuron.

From this condition, we can calculate the amount of update $\triangle v_i$ required as follows

$$\begin{aligned} u_i(t+1) &= \sum_{j=1}^{N_I} w_{ij} v_j + w_{ii} \triangle v_i + b_i \\ &= u_i(t) + w_{ii} \triangle v_i \\ &= 0 \end{aligned}$$

which implies

$$\triangle v_i = -\frac{u_i(t)}{w_{ii}} \tag{3.11}$$

and the final output of the neuron is again given by

$$v_i(t+1) = v_i(t) + \triangle v_i(t) \tag{3.12}$$

It can be shown that the point $v_i(t+1)$ is the global minimum of the energy function $E(v_i)$ by evaluating the second derivative of E as follows:

$$\begin{aligned} \frac{\partial^2 E}{\partial v_i^2} &= -w_{ii} \\ &= \sum_{p=1}^{N_I} h_{pi}^2 + \lambda \sum_{p=1}^{N_I} d_{pi}^2 > 0 \end{aligned}$$

for positive λ. As a result, we can attain this global minimum by equation (3.11) in a single step [37] rather than restricting the update values to ± 1 as in Zhou *et.al* [118] . In addition, equation (3.11) expresses the change in neuron output as a function of the regularization parameter λ through w_{ii}. This dependency is essential for the reformulation of the current adaptive regularization problem into a learning problem for a set of model-based neurons.

3.3 Adaptive Regularization–An Alternative Formulation

In this section we propose an alternative formulation of adaptive regularization which centers on the concept of *regularization parameters as model-based neuronal weights.* By adopting this alternative viewpoint, a neat correspondence is found to exist between the mechanism of adaptive regularization and the computation of an artificial model-based neuron.

Referring to equation (3.11) and identifying the gray-level value $\hat{x}_i(t)$ at the lexicographically ordered pixel site i with the neuron output $v_i(t)$, we can express the instantaneous restoration stepsize $\triangle v_i(t)$ in (3.11) in the form of a neural computational process as follows

$$\triangle v_i = -\frac{u_i(t)}{w_{ii}}$$

$$
\begin{aligned}
&= -\frac{\sum_{j=1}^{N_I} w_{ij} v_j + b_i}{w_{ii}} \\
&= \sum_{j=1}^{N_I} p_{ij} v_j + q_i b_i \\
&= \mathbf{p}_i^T \mathbf{v}_i
\end{aligned} \tag{3.13}
$$

where

$$
\begin{aligned}
\mathbf{p}_i &= [p_{i1}, \ldots, p_{iN_I}, q_i]^T \in \mathbf{R}^{N_I+1} \\
\mathbf{v}_i &= [v_1, \ldots, v_{N_I}, b_i]^T \in \mathbf{R}^{N_I+1}
\end{aligned}
$$

and the weights of this hypothetical neuron are defined as

$$p_{ij} = -\frac{w_{ij}}{w_{ii}} \tag{3.14}$$

$$q_i = -\frac{1}{w_{ii}} \tag{3.15}$$

The weights w_{ij} of the primary Hopfield neuron (as opposed to those of this hypothetical neuron) are in turn given by equation (3.3)

$$
\begin{aligned}
w_{ij} &= -\sum_{p=1}^{N_I} h_{pi} h_{pj} - \lambda \sum_{p=1}^{N_I} d_{pi} d_{pj} \\
&= g_{ij} + \lambda l_{ij}
\end{aligned} \tag{3.16}
$$

where we define

$$g_{ij} = -\sum_{p=1}^{N_I} h_{pi} h_{pj} \tag{3.17}$$

$$l_{ij} = -\sum_{p=1}^{N_I} d_{pi} d_{pj} \tag{3.18}$$

From these equations, the weights p_{ij} of the hypothetical neuron can be expressed as a function of the regularization parameter λ as follows

$$p_{ij} = -\frac{g_{ij} + \lambda l_{ij}}{g_{ii} + \lambda l_{ii}} \tag{3.19}$$

$$q_i = -\frac{1}{g_{ii} + \lambda l_{ii}} \tag{3.20}$$

To re-interpret these equations as the computation of a model-based neuron, we re-cast them into its two-dimensional form from the previous lexicographical ordered form. Defining the *lexicographical mapping* $\mathcal{L}$ as follows

$$i = \mathcal{L}(i_1, i_2) = i_1 N_x + i_2 \tag{3.21}$$

where (i_1, i_2) is the corresponding 2-D position of the i-th lexicographical vector entry in the image lattice, and N_x is the width of the image lattice. With this mapping, we can re-interpret equation (3.13) in a 2-D setting

$$\triangle\widetilde{v}_{i_1,i_2} = \sum_{k=-L_c}^{L_c} \sum_{l=-L_c}^{L_c} \widetilde{p}_{i_1,i_2,k,l}\widetilde{v}_{i_1+k,i_2+l} + \widetilde{q}_{i_1,i_2}\widetilde{b}_{i_1,i_2} \tag{3.22}$$

where the tilded form of the variables indicate that the current quantity is indexed by its 2-D position in the lattice, rather than its 1-D position in the lexicographically ordered vector. The summation in the equation is taken over the support of a *2-D neuronal weight mask*, the size of which depends on the extent of the original point spread function [118] and which is $(2L_c + 1)^2$ in this case. For $i = \mathcal{L}(i_1, i_2)$, we define

$$\widetilde{v}_{i_1,i_2} = v_i \tag{3.23}$$

$$\widetilde{p}_{i_1,i_2,k,l} = p_{i,\mathcal{L}(i_1+k,i_2+l)} \tag{3.24}$$

and the variables $\triangle\widetilde{v}_{i_1,i_2}$, $\widetilde{q}_{i_1,i_2}$ and $\widetilde{b}_{i_1,i_2}$ are similarly defined as $\widetilde{v}_{i_1,i_2}$. For spatially invariant degradation, the variables $\widetilde{p}_{i_1,i_2,k,l}$ and $\widetilde{q}_{i_1,i_2}$ are independent of the position (i_1, i_2) in the image lattice, and the above equation can be re-written as

$$\begin{aligned} \triangle\widetilde{v}_{i_1,i_2} &= \sum_{k=-L_c}^{L_c} \sum_{l=-L_c}^{L_c} \widetilde{p}_{k,l}\widetilde{v}_{i_1+k,i_2+l} + \widetilde{q}\widetilde{b}_{i_1,i_2} \\ &= \widetilde{\mathbf{p}}^T\widetilde{\mathbf{v}}_{i_1,i_2} \end{aligned} \tag{3.25}$$

where

$$\begin{aligned} \widetilde{\mathbf{p}} &= [\widetilde{p}_{-L_c,-L_c}, \cdots, \widetilde{p}_{0,0}, \cdots, \widetilde{p}_{L_c,L_c}, \widetilde{q}]^T \in \mathbf{R}^{N_c+1} \\ \widetilde{\mathbf{v}}_{i_1,i_2} &= [\widetilde{v}_{i_1-L_c,i_2-L_c}, \cdots, \widetilde{v}_{i_1,i_2}, \cdots, \widetilde{v}_{i_1+L_c,i_2+L_c}, \widetilde{b}_{i_1,i_2}]^T \in \mathbf{R}^{N_c+1} \end{aligned}$$

and $N_c = (2L_c + 1)^2$.

3.3.1 Correspondence with the General HMBNN Architecture

For spatially invariant degradation with small support, we have the condition $N_c \ll N_I$, where N_I is the number of pixels in the image, and we can view equation (3.25) as a local convolution operation over a selected neighborhood of $\widetilde{v}_{i_1,i_2}$. On the other hand, due

to the invariant nature of $\widetilde{p}_{k,l}$ and $\widetilde{q}$ for all i_1 and i_2, this operation can alternatively be viewed as the computational process of a model-based neuron with input vector $\widetilde{\mathbf{v}}_{i_1,i_2}$ and weight vector $\widetilde{\mathbf{p}}$. In other words, if we assume that the image is subdivided into regions $\mathcal{R}_r, r = 1, \ldots, R$, and we assign a sub-network with a single model-based neuron to each region, i.e., $S_r = 1$ for each r according to the notation in Chapter 2, then the local model-based neuron output corresponding to $\mathcal{R}_r$ can be represented as

$$\psi_r(\tilde{\mathbf{v}}_{i_1,i_2}, \tilde{\mathbf{p}}_{k,l}(\lambda_r)) = \tilde{\mathbf{p}}_{k,l}^T(\lambda_r)\tilde{\mathbf{v}}_{i_1,i_2} \tag{3.26}$$

where λ_r, the regional regularization parameter, can be considered the *scalar* model-based weight associated with each region $\mathcal{R}_r$. Since there is no competitive learning involved in the operation of this restoration network , it can be classified within the class of hidden-node hierarchical networks as described in Chapter 2. The architecture of this HMBNN for adaptive regularization is shown in Figure 3.1.

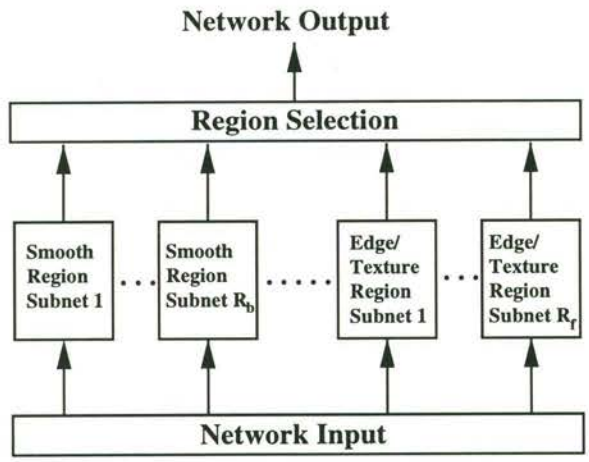


Figure 3.1: The model-based neural network with hierarchical architecture for adaptive regularization

Adopting the weight-parameterized model-based neuron in Chapter 2 due to the natural way in which $\lambda_r \in \mathbf{R}$ is embedded in the high-dimensional weight vector $\tilde{\mathbf{p}}_{k,l}(\lambda_r) \in \mathbf{R}^{N_c+1}$, and referring to equation (3.19), we can obtain the component mappings $\mathcal{M}_n$ of the model-based neuron for the r-th sub-network as follows

$$\widetilde{p}_{k,l}(\lambda_r) \;=\; \mathcal{M}_n(\lambda_r) = -\frac{\widetilde{g}_{k,l} + \lambda_r \widetilde{l}_{k,l}}{\widetilde{g}_{0,0} + \lambda_r \widetilde{l}_{0,0}} \qquad n = 1, \ldots, N_c \tag{3.27}$$

$$\tilde{q}(\lambda_r) \;=\; \mathcal{M}_{N_c+1}(\lambda_r) = -\frac{1}{\tilde{g}_{0,0} + \lambda_r \tilde{l}_{0,0}} \tag{3.28}$$

where the mapping index n corresponds to some specific arrangement of the indices (k, l) in the 2-D weight mask $\tilde{p}_{k,l}$, and the values of $\tilde{g}_{k,l}$ and $\tilde{l}_{k,l}$ can be obtained from their lexicographical counterparts g_{ij} and l_{ij} using equation (3.24). The resulting concatenation $\mathcal{M}$ of the components is thus a mapping from $\mathbf{R}$ to $\mathbf{R}^{N_c+1}$. Corresponding to the notation of section 2.2, we have $M = 1$ and $N = N_c + 1$ for the current adaptive regularization problem. In addition, each mapping is in the form of a *first-order rational polynomial* of λ_r, the differentiability of which ensures that the weights $\tilde{p}_{k,l}$ are trainable. The structure of the model-based neuron adapted for restoration is shown in Figure 3.2.

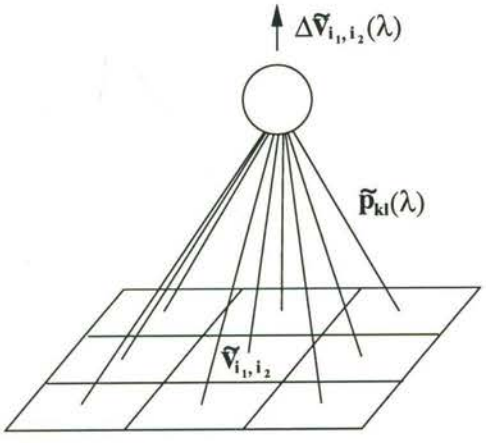


Figure 3.2: The model-based neuron for adaptive regularization

To complete the formulation, we define a training set for the current model-based neuron. Assuming that for a certain region $\mathcal{R}_r$ in the image lattice, there exists a desired restoration behavior $\Delta\tilde{v}^d_{i_1,i_2}$ for each pixel $(i_1, i_2) \in \mathcal{R}_r$. We can then define the training set $\mathcal{V}_r$ as follows

$$\mathcal{V}_r = \{(\tilde{\mathbf{v}}_{i_1,i_2}, \Delta\tilde{v}^d_{i_1,i_2}) : (i_1, i_2) \in \mathcal{R}_r\} \tag{3.29}$$

where the input vector $\tilde{\mathbf{v}}_{i_1,i_2}$ contains the gray-level values of pixels in the N_c-neighborhood of (i_1, i_2). Defining the instantaneous cost function $\mathcal{E}_t$ of the neuron as follows,

$$\mathcal{E}_t = \frac{1}{2}(\Delta\tilde{v}^d_{i_1,i_2} - \Delta\tilde{v}_{i_1,i_2})^2 \tag{3.30}$$

we can then apply steepest descent to modify the value of the regularization parameter λ_r.

$$\lambda_r(t+1) = \lambda_r(t) - \eta \frac{\partial \mathcal{E}_t}{\partial \lambda_r} \tag{3.31}$$

where

$$\frac{\partial \mathcal{E}_t}{\partial \lambda_r} = -(\triangle \widetilde{v}^d_{i_1,i_2} - \triangle \widetilde{v}_{i_1,i_2}) \frac{\partial (\triangle \widetilde{v}_{i_1,i_2})}{\partial \lambda_r} \tag{3.32}$$

To evaluate the partial derivative in equation (3.32), we first define the following two quantities

$$\widetilde{\alpha}_{i_1,i_2} = \sum_{k=-L_c}^{L_c} \sum_{l=-L_c}^{L_c} \widetilde{g}_{k,l} \widetilde{v}_{i_1+k,i_2+l} \tag{3.33}$$

$$\widetilde{\beta}_{i_1,i_2} = \sum_{k=-L_c}^{L_c} \sum_{l=-L_c}^{L_c} \widetilde{l}_{k,l} \widetilde{v}_{i_1+k,i_2+l} \tag{3.34}$$

From equation (3.25), the change in neuron output, $\triangle \widetilde{v}_{i_1,i_2}$, is then expressed in terms of these quantities as follows,

$$\begin{aligned} \triangle \widetilde{v}_{i_1,i_2} &= \sum_{k=-L_c}^{L_c} \sum_{l=-L_c}^{L_c} \widetilde{p}_{k,l} \widetilde{v}_{i_1+k,i_2+l} + \widetilde{q} \widetilde{b}_{i_1,i_2} \\ &= -\frac{\sum_{k=-L_c}^{L_c} \sum_{l=-L_c}^{L_c} (\widetilde{g}_{k,l} + \lambda_r \widetilde{l}_{k,l}) \widetilde{v}_{i_1+k,i_2+l} + \widetilde{b}_{i_1,i_2}}{\widetilde{g}_{0,0} + \lambda_r \widetilde{l}_{0,0}} \\ &= -\frac{\widetilde{\alpha}_{i_1,i_2} + \lambda_r \widetilde{\beta}_{i_1,i_2} + \widetilde{b}_{i_1,i_2}}{\widetilde{g}_{0,0} + \lambda_r \widetilde{l}_{0,0}} \end{aligned} \tag{3.35}$$

From this equation, we can evaluate the derivative in equation (3.32)

$$\begin{aligned} \frac{\partial (\triangle \widetilde{v}_{i_1,i_2})}{\partial \lambda_r} &= \frac{\partial}{\partial \lambda_r} \left(-\frac{\widetilde{\alpha}_{i_1,i_2} + \lambda_r \widetilde{\beta}_{i_1,i_2} + \widetilde{b}_{i_1,i_2}}{\widetilde{g}_{0,0} + \lambda_r \widetilde{l}_{0,0}} \right) \\ &= \frac{(\widetilde{\alpha}_{i_1,i_2} + \widetilde{b}_{i_1,i_2}) \widetilde{l}_{0,0} - \widetilde{\beta}_{i_1,i_2} \widetilde{g}_{0,0}}{(\widetilde{g}_{0,0} + \lambda_r \widetilde{l}_{0,0})^2} \end{aligned} \tag{3.36}$$

We can see from equation (3.36) that the evaluation of the derivatives depend on variables which have already been pre-computed for the purpose of the primary restoration. For example, the weights $\widetilde{g}_{0,0}$, $\widetilde{l}_{0,0}$ and the bias $\widetilde{b}_{i_1,i_2}$ are pre-computed at the beginning of the restoration process, and the quantities $\widetilde{\alpha}_{i_1,i_2}$ and $\widetilde{\beta}_{i_1,i_2}$ are already evaluated for the purpose of determining the instantaneous pixel value change $\triangle \widetilde{v}_{i_1,i_2}$ in equation (3.25). As a result, the adaptive regularization does not involve excessive computational overhead.

The size of the training set $\mathcal{V}_r$ depends on the extent of the region $\mathcal{R}_r$ in definition (3.29). It is possible to define $\mathcal{R}_r$ to include the entire image lattice, which would amount to a fixed regularization scheme where we will search for the optimum global regularization parameter value using a single model-based sub-network, but for the problem of adaptive regularization, the region $\mathcal{R}_r$ is usually restricted to a subset of the image lattice, and several such regions are defined for the entire lattice to form an image partition. We associate each such region with a sub-network, the totality of which forms a model-based neural network with hierarchical architecture. In general, we would expect the emergence of regional λ_r values which results in the improved visual quality for the associated region through the training process. This in turn depends critically on the definition of our desired output $\triangle\tilde{v}^d_{i_1,i_2}$ in the regional training set and our image partition. These two issues will be addressed in Sections 3.4 and 3.5.

3.4 Regional Training Set Definition

To complete the definition of the regional training set $\mathcal{V}_r$, we should supply the desired output $\triangle\tilde{v}^d_{i_1,i_2}$ for each input vector $\tilde{\mathbf{v}}_{i_1,i_2}$ in the set. The exact value of $\triangle\tilde{v}^d_{i_1,i_2}$ which would lead to an optimal visual quality for the particular region concerned is normally unknown due to the usually unsupervised nature of image restoration problems. Nevertheless, an appropriate approximation of this value can usually be obtained by employing a *neighborhood-based prediction scheme* to estimate $\triangle\tilde{v}^d_{i_1,i_2}$ for the current pixel. This is in the same spirit as the neighborhood-based estimation technique widely used in non-linear filtering applications where a nonlinear function defined on a specified neighborhood of the current pixel is used to recover the correct gray-level value from its noise-corrupted value [87] . The nonlinear filters are usually designed with the purpose of noise suppression in mind. The resulting operations thus have a tendency to over-smooth the edge and textured regions of the image. Remedies to this over-smoothing problem include various edge adaptive filtering schemes where the prediction is performed along the current edge orientation to avoid filtering across the edges [5, 87] . In this work, we would similarly

adopt two different prediction schemes for the smooth and textured regions and use the resulting estimated values as the desired outputs for the respective regional training sets.

For combined edge/textured regions, we shall adopt a prediction scheme which emphasizes the dynamic range of the associated region, or equivalently, a scheme which biases towards large values of $\triangle\tilde{v}^d_{i_1,i_2}$, but at the same time suppresses excessive noise occurring in those regions. For the edge/texture prediction scheme, the following *prediction neighborhood set* for each pixel was adopted.

$$\mathcal{N}_p = \{(k,l) : k,l = -L_p, \ldots, 0, \ldots, L_p\} \tag{3.37}$$

In addition, we define the following mean gray level value $\overline{\tilde{v}}_{i_1,i_2}$ with respect to this neighborhood set as follows:

$$\overline{\tilde{v}}_{i_1,i_2} = \frac{1}{N_p} \sum_{k=-L_p}^{L_p} \sum_{l=-L_p}^{L_p} \tilde{v}_{i_1+k,i_2+l} \tag{3.38}$$

where $\tilde{v}_{i_1,i_2}$ denotes the gray-level value at (i_1, i_2) and $N_p = (2L_p + 1)^2$.

To avoid the problem of over-smoothing in the combined edge/textured regions while suppressing excessive noise at the same time, we have adopted the concept of *weighted order statistic (WOS) filter* [19, 92] in deriving a suitable desired network output $\triangle\tilde{v}^d_{i_1,i_2}$ for the training set. The set of order statistics corresponding to the prediction neighborhood set can be defined as follows: for the n_p-th order statistic $\tilde{v}^{(n_p)}_{i_1,i_2}$

$$n_p = P(k,l) \quad \exists (k,l) \in \mathcal{N}_p \tag{3.39}$$

where $P : \mathcal{N}_p \longrightarrow \{1, \ldots, N_p\}$ is a one-to-one mapping such that the following condition is satisfied:

$$\tilde{v}^{(1)}_{i_1,i_2} \leq \ldots \leq \tilde{v}^{(n_p)}_{i_1,i_2} \leq \ldots \leq \tilde{v}^{(N_p)}_{i_1,i_2} \tag{3.40}$$

The output of a weighted order statistic filter is defined as the linear combination of the order statistics

$$\tilde{v}^d_{i_1,i_2} = \sum_{n_p=1}^{N_p} \omega^{(n_p)} \tilde{v}^{(n_p)}_{i_1,i_2} \tag{3.41}$$

For odd N_p, the simplest example of a WOS filter is the median filter [87] where

$$\begin{aligned} \omega^{(M_p)} &= 1 \\ \omega^{(n_p)} &= 0 \qquad n_p \neq M_p \end{aligned} \tag{3.42}$$

and

$$M_p = \frac{N_p + 1}{2} \tag{3.43}$$

Therefore, the WOS filter can be considered as a generalization of the median filter where the information from all the order statistics are combined to provide an improved estimate of a variable. In general, for the purpose of noise filtering, the filter weights $\omega^{(1)}$ and $\omega^{(N_p)}$ are chosen such that $\omega^{(1)} \approx 0$ and $\omega^{(N_p)} \approx 0$, since the corresponding order statistics $\tilde{v}_{i_1,i_2}^{(1)}$ and $\tilde{v}_{i_1,i_2}^{(N_p)}$ usually represent outliers.

Adopting the value $L_p = 1$ and $N_p = (2 \cdot 1 + 1)^2 = 9$ as the size of the prediction neighborhood set , we define the *predicted gray level value* $\tilde{v}_{i_1,i_2}^d$ and the corresponding desired network output $\triangle\tilde{v}_{i_1,i_2}^d$ according to the operation of a WOS filter as follows:

$$\tilde{v}_{i_1,i_2}^d = \begin{cases} \tilde{v}_{i_1,i_2}^{(3)} & \tilde{v}_{i_1,i_2} < \overline{\tilde{v}}_{i_1,i_2} \\ \tilde{v}_{i_1,i_2}^{(7)} & \tilde{v}_{i_1,i_2} \geq \overline{\tilde{v}}_{i_1,i_2} \end{cases} \tag{3.44}$$

$$\triangle\tilde{v}_{i_1,i_2}^d = \tilde{v}_{i_1,i_2}^d - \tilde{v}_{i_1,i_2} \tag{3.45}$$

or equivalently

$$\tilde{v}_{i_1,i_2}^d = \sum_{n_p=1}^{9} \omega^{(n_p)} \tilde{v}_{i_1,i_2}^{(n_p)} \tag{3.46}$$

where

$$\omega^{(3)} = 1, \omega^{(n_p)} = 0, n_p \neq 3 \quad \text{for } \tilde{v}_{i_1,i_2} < \overline{\tilde{v}}_{i_1,i_2}$$

$$\omega^{(7)} = 1, \omega^{(n_p)} = 0, n_p \neq 7 \quad \text{for } \tilde{v}_{i_1,i_2} \geq \overline{\tilde{v}}_{i_1,i_2}$$

The motivation for choosing the respective order statistics for the two different cases is that, for the case $\tilde{v}_{i_1,i_2} \geq \overline{\tilde{v}}_{i_1,i_2}$, we assume that the true gray level value lies in the interval $[\overline{\tilde{v}}_{i_1,i_2}, \tilde{v}_{i_1,i_2}^{(9)}]$. For blurred or partially restored image, this corresponds approximately to the interval $[\tilde{v}_{i_1,i_2}^{(5)}, \tilde{v}_{i_1,i_2}^{(9)}]$ with its endpoints at the median and maximum gray level values.

Within this interval, we cannot choose $\tilde{v}_{i_1,i_2}^d = \tilde{v}_{i_1,i_2}^{(5)} \approx \overline{\tilde{v}}_{i_1,i_2}$, as this will result in excessive smoothing for the combined edge/textured regions. Neither can we choose $\tilde{v}_{i_1,i_2}^d = \tilde{v}_{i_1,i_2}^{(9)}$ with the corresponding order statistic being usually considered an outlier which does not accurately reflect the correct gray level value of the current pixel. A possible candidate could be $\tilde{v}_{i_1,i_2}^d = 0.5(\tilde{v}_{i_1,i_2}^{(5)} + \tilde{v}_{i_1,i_2}^{(9)})$, but the presence of the outlier $\tilde{v}_{i_1,i_2}^{(9)}$

in the linear combination will still result in non-representative predictions, especially for high levels of noise. To ensure the comparative noise immunity of the resulting estimate while avoiding excessive smoothing, the choice $\widetilde{v}^d_{i_1,i_2} = \widetilde{v}^{(7)}_{i_1,i_2}$ represents a compromise which offers the additional advantage that the gray level value is among one of those in the prediction neighborhood set. The adoption of this value thus implicitly imposes a continuity constraint between the current and the neighboring pixels. The choice of $\widetilde{v}^d_{i_1,i_2} = \widetilde{v}^{(3)}_{i_1,i_2}$ for the case $\widetilde{v}_{i_1,i_2} < \overline{\widetilde{v}}_{i_1,i_2}$ is similarly justified.

On the other hand, we should adopt a prediction scheme which biases towards small values of $\triangle\widetilde{v}^d_{i_1,i_2}$ for the smooth regions to suppress the more visible noise and ringing there. In view of this, the following prediction scheme is adopted for the smooth regions

$$\widetilde{v}^d_{i_1,i_2} = \overline{\widetilde{v}}_{i_1,i_2} \tag{3.47}$$

$$\triangle\widetilde{v}^d_{i_1,i_2} = \widetilde{v}^d_{i_1,i_2} - \widetilde{v}_{i_1,i_2} \tag{3.48}$$

This prediction scheme essentially employs the local mean, which serves as a useful indicator of the correct gray-level values in smooth regions, as an estimate for the current gray-level value. Alternatively, it can be viewed as the operation of a filter mask with all its coefficients being $\frac{1}{N_p}$.

The essential difference between the current approach and traditional adaptive nonlinear filtering techniques [87] is that, whereas the traditional filtering techniques *replace* the current gray-level value with the above predicted value, the current scheme use this predicted value as a training *guidance* by incorporating it as the desired output in the regional training set. We then apply steepest descent to change the corresponding regional regularization parameter according to the information in these training patterns. In this way, both the information of the degradation mechanism (in the form of the model-based neuronal weights $\widetilde{p}_{k,l}$) and the regional image model (in the form of the regional training set $\mathcal{V}_r$) are exploited to achieve a more accurate restoration.

3.5 Determination of the Image Partition

Conforming to the above description of an image as a combination of edge/textured and smooth components, we denote the edge/textured components by $\mathcal{F}_{r_f}, r_f = 1, \ldots, R_f$ and the smooth components by $\mathcal{B}_{r_b}, r_b = 1, \ldots, R_b$. We further define the following combinations of these components

$$\mathcal{F} = \bigcup_{r_f} \mathcal{F}_{r_f} \tag{3.49}$$

$$\mathcal{B} = \bigcup_{r_b} \mathcal{B}_{r_b} \tag{3.50}$$

$$\mathcal{P}_F = \{\mathcal{F}_{r_f}, r_f = 1, \ldots, R_f\} \tag{3.51}$$

$$\mathcal{P}_B = \{\mathcal{B}_{r_b}, r_b = 1, \ldots, R_b\} \tag{3.52}$$

$$\mathcal{P}_R = \mathcal{P}_F \cup \mathcal{P}_B \tag{3.53}$$

Our partitioning strategy is to first classify each pixel as belonging to the region $\mathcal{F}$ or $\mathcal{B}$ and then derive the partitions $\mathcal{P}_F$ and $\mathcal{P}_B$ using connectivity criteria. We perform the preliminary classification by adopting the following local activity measure δ_{i_1,i_2} for each pixel

$$\delta_{i_1,i_2} = \ln(\sigma_{i_1,i_2}) \tag{3.54}$$

where

$$\sigma_{i_1,i_2} = \left(\frac{1}{N_p} \sum_{k=-L_p}^{L_p} \sum_{l=-L_p}^{L_p} (\widetilde{v}_{i_1+k,i_2+l} - \overline{\widetilde{v}}_{i_1,i_2})^2\right)^{\frac{1}{2}} \tag{3.55}$$

and $\overline{\widetilde{v}}_{i_1,i_2}$ is defined in equation (3.38). The logarithm mapping is adopted to approximate the non-linear operation in the human vision system which transforms intensity values to perceived contrasts [50]. We then assign the current pixel to either $\mathcal{F}$ or $\mathcal{B}$ according to the value of δ_{i_1,i_2} relative to a threshold T

$$\mathcal{F} = \{(i_1, i_2) : \delta_{i_1,i_2} > T\} \tag{3.56}$$

$$\mathcal{B} = \{(i_1, i_2) : \delta_{i_1,i_2} \leq T\} \tag{3.57}$$

The threshold T is usually selected according to some optimality criteria based on the particular image content. In this work, we have chosen $T = T^*$, where T^* is the threshold

which minimizes the total within-class variance ϱ^2 of δ_{i_1,i_2}, i.e.,

$$T^* = \arg\min_T \varrho^2(T) \tag{3.58}$$

where

$$\varrho^2 = \frac{1}{|\mathcal{B}|} \sum_{(i_1,i_2)\in\mathcal{B}} (\sigma_{i_1,i_2} - \sigma_{\mathcal{B}})^2 + \frac{1}{|\mathcal{F}|} \sum_{(i_1,i_2)\in\mathcal{F}} (\sigma_{i_1,i_2} - \sigma_{\mathcal{F}})^2 \tag{3.59}$$

and

$$\sigma_{\mathcal{B}} = \frac{1}{|\mathcal{B}|} \sum_{(i_1,i_2)\in\mathcal{B}} \sigma_{i_1,i_2} \tag{3.60}$$

$$\sigma_{\mathcal{F}} = \frac{1}{|\mathcal{F}|} \sum_{(i_1,i_2)\in\mathcal{F}} \sigma_{i_1,i_2} \tag{3.61}$$

The partitions $\mathcal{P}_F$ and $\mathcal{P}_B$ are in turn extracted from the sets $\mathcal{F}$ and $\mathcal{B}$ by considering each element of $\mathcal{P}_F$ or $\mathcal{P}_B$ to be a maximally connected component of $\mathcal{F}$ or $\mathcal{B}$. To be more precise, if we adopt the usual 8-path connectivity relation $\mathcal{C}_8$ as our connectivity criterion in the case of $\mathcal{F}$ [36], then the partition $\mathcal{P}_F$ will be given by the *quotient set* $\mathcal{F}/\mathcal{C}_8$ of $\mathcal{F}$ under the equivalence relation $\mathcal{C}_8$. The partition $\mathcal{P}_B$ is similarly defined.

In view of the simplicity of this preliminary segmentation scheme and the fact that the relative proportion of the two region types would vary during the course of image restoration, we should adopt some re-segmentation procedures throughout the restoration process to account for this variation. In this work, we have used a modified version of the *nearest neighbor classification procedure* [103] to perform the re-assignment, and we have restricted the re-assignment to pixels on the region boundary. The following *neighboring region set* $\mathcal{N}^{\mathcal{R}}_{i_{b1},i_{b2}} \subset \mathcal{P}_R$ is defined for each such boundary pixel

$$\mathcal{N}^{\mathcal{R}}_{i_{b1},i_{b2}} = \{\mathcal{R}_q, q = 1, \ldots, Q\} \tag{3.62}$$

where each region $\mathcal{R}_q$ in the set is adjacent to the boundary pixel (i_{b1}, i_{b2}), or more precisely, there exists at least one pixel in each $\mathcal{R}_q$ which is in the 8-neighborhood of (i_{b1}, i_{b2}). Corresponding to each $\mathcal{R}_q$ we can define the following regional activity $\overline{\sigma}_q$

$$\overline{\sigma}_q = \frac{1}{|\mathcal{R}_q|} \sum_{(i_1,i_2)\in\mathcal{R}_q} \sigma_{i_1,i_2} \tag{3.63}$$

where σ_{i_1,i_2} is defined in equation (3.55). With the definition of these variables, we can proceed with the re-classification of the boundary pixels by adopting the following nearest neighbor decision rule for (i_{b1}, i_{b2})

$$(i_{b1}, i_{b2}) \in \mathcal{R}_{q^*} \quad \text{if } |\sigma_{i_{b1},i_{b2}} - \overline{\sigma}_{q^*}| < |\sigma_{i_{b1},i_{b2}} - \overline{\sigma}_q|, q = 1, \ldots, Q \tag{3.64}$$

The application of this decision rule is manifested as a continual change in the boundary of each region in such a way as to increase the homogeneity of the activities of the regions. Finally, with the full determination of the various texture and smooth components in the image by the above scheme, we can apply the steepest descent rule (3.31) to each regional regularization parameter λ_r and using the prediction schemes for the respective regions to achieve adaptive regularization.

3.6 Experimental Results

The current algorithm was applied to a number of images under various conditions of degradations. The batch of images, which includes one depicting a flower, a woman's face (Lena) and an eagle respectively, are particularly relevant for testing the efficacy of the current scheme, as they exhibit a combination of smooth backgrounds together with a wide variety of textural patterns. For example, in the Lena image, we can contrast the relatively fine textures in the feathers on the hat with the almost featureless background and face, and on the flower image, we can compare the smooth petal regions with the textured region consisting of the stamen. These images are shown in Figure 3.3.

We applied progressive levels of degradation to the flower's image to evaluate the robustness of the current algorithm. The different degrees of degradation applied include, in increasing order of severity, a 5 × 5 Gaussian PSF (point spread function) with standard deviation σ_g equals 1 and with Gaussian noise at the level of 30dB BSNR (Blurred Signal to Noise Ratio) [55, 67] added, a 5 × 5 uniform PSF with 30dB level noise added, and a 5 × 5 uniform PSF with 20dB level noise added, representing the most severe of the three

(a)

(b)

(c)

Figure 3.3: Images used in the adaptive regularization experiments. (a) Flower image (b) Lena image (c) Eagle image

conditions . The blurred signal to noise ratio is defined as follows

$$BSNR = 10 \log_{10} \frac{\sigma^2_{\mathbf{Hx}}}{\sigma^2_{\mathbf{n}}} \tag{3.65}$$

where $\sigma^2_{\mathbf{Hx}}$ is the variance of the blurred image, and $\sigma^2_{\mathbf{n}}$ is the variance of the additive noise.

In addition, we have applied the 5 × 5 uniform PSF with 30dB noise to the other two images to evaluate the algorithm's performance for images with different texture/background compositions.

We first applied the algorithm to the flower image under the 5 × 5 Gaussian blur. The results are shown in Figure 3.4 . For the purpose of comparison, we have included the results of the non-adaptive version of the Hopfield network image restoration algorithm proposed by Zhou *et al* [118]. The degraded image is shown in Figure 3.4(a), and the non-adaptive image restoration results are shown in Figures 3.4(b) and (c) respectively. For Figure 3.4(b), we have used a small regularization parameter $\lambda = 0.0005$ for the whole image. We can notice the noisy appearance of the background regions due to the inadequate suppression of noises in those regions, although the effect is not serious due to the narrow width of the PSF which makes the restoration problem only moderately ill-conditioned. In Figure 3.4(c), we have increased the global regularization parameter to $\lambda = 0.004$ to suppress the noises and at the same time preserve the features. The adaptive result using the current algorithm is shown in Figure 3.4(d). Comparing this with the under-regularized case in Figure 3.4(b), we can observe that, with the adoption of multiple regularization parameters and the optimization of their values using the current algorithm, the noises in the smooth regions can be adequately suppressed. However, for the current case, the non-adaptive result in Figure 3.4(c), where the regularization parameter is judiciously chosen, is comparable with the adaptive result due to the narrow width of the PSF.

The results for the 5×5 uniform PSF with 30dB noise added are shown in Figure 3.5. This PSF characterizes a more severe degradation, which is evident from the appearance of the degraded image in Figure 3.5(a), and the inversion of this degradation represents

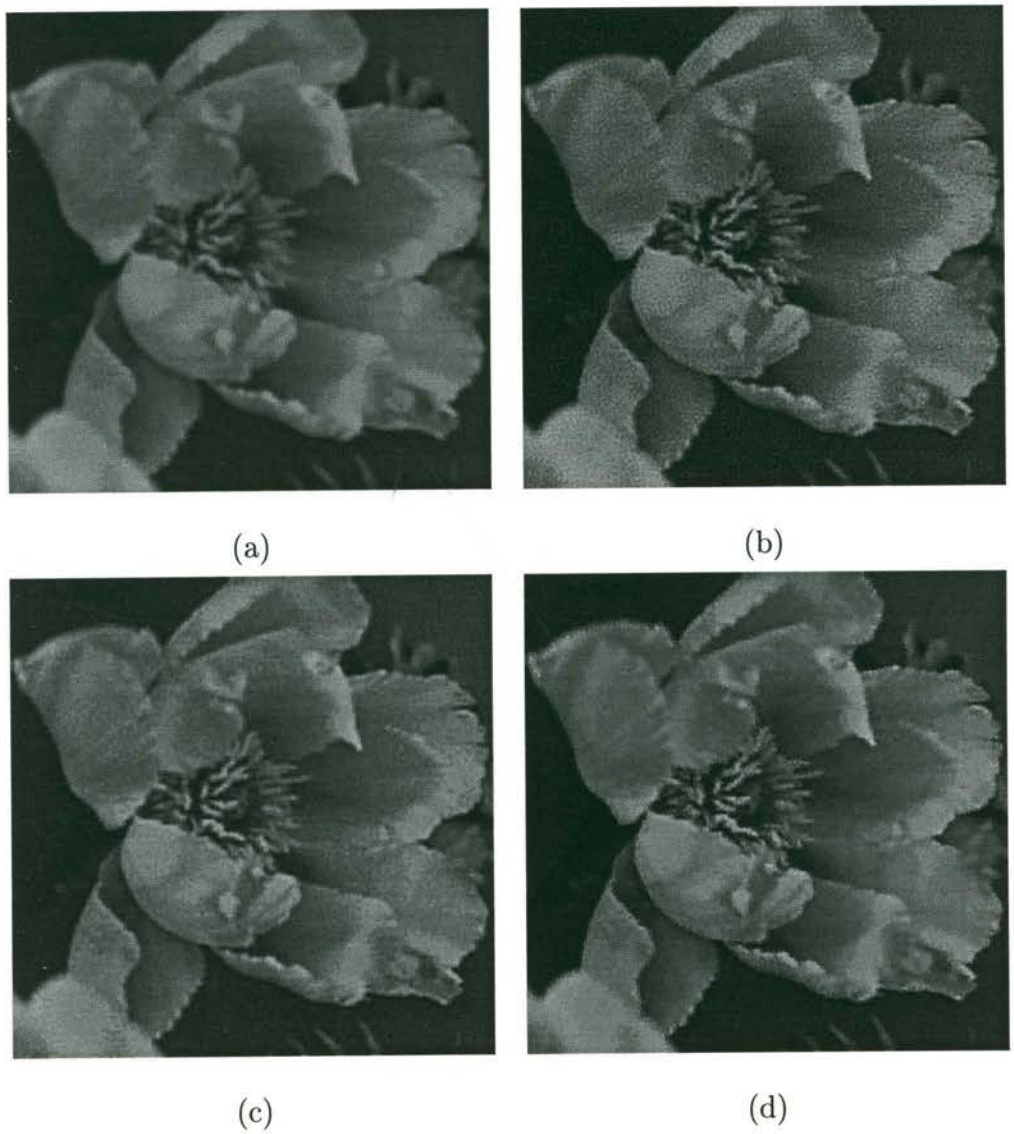


Figure 3.4: Restored Flower images (5 × 5 Gaussian blur, 30dB BSNR). (a) Blurred image. (b)-(d) Restored images using (b) non-adaptive Hopfield restoration algorithm (small λ). (c) non-adaptive Hopfield restoration algorithm (optimally adjusted λ by user) (d) HMBNN.

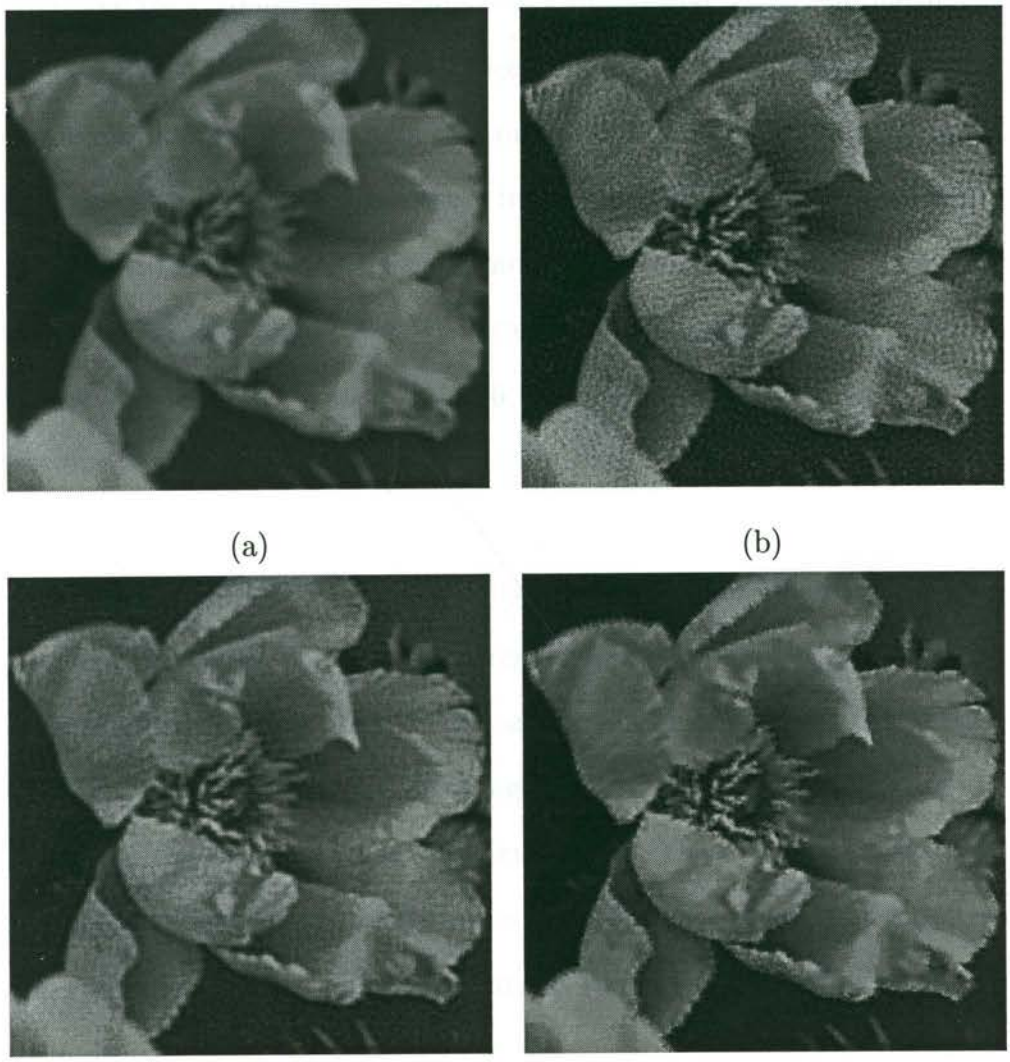


Figure 3.5: Restored Flower images (5×5 uniform blur, 30dB BSNR). (a) Blurred image. (b)-(d) Restored images using (b) non-adaptive Hopfield restoration algorithm (small λ). (c) non-adaptive Hopfield restoration algorithm (optimally adjusted λ by user). (d) HMBNN.

a more ill-conditioned problem due to the appearance of periodic zeros in the Fourier transform of the PSF [3]. This is immediately apparent from the increased levels of amplified noises in the smooth regions for the non-adaptive restoration results using a small global λ ($\lambda = 0.0005$) in Figure 3.5(b). In Figure 3.5(c), we have set $\lambda = 0.004$ in order that the background noise is effectively suppressed while the image features are adequately preserved. However, we can notice that, due to the non-adaptive nature of the process, some residual noises remain in the smooth regions while blurring of the edges and textures can be observed. Comparing these images with HMBNN result in Figure 3.5(d), we can notice that the background noise is more effectively suppressed. Some of the edges and textures are correctly regularized while some of them appear blurred. These problems will be further discussed in chapters 5 and 6.

The results for the 5×5 uniform PSF with 20dB BSNR are shown in Figure 3.6 . Whereas the description for the images in Figure 3.5 is mostly applicable to the corresponding images in Figure 3.6 , the difficulty in restoring the images under this PSF is apparent. One can notice the further increased noise levels in the smooth regions for the under-regularized case in Figure 3.6(b), and any attempts to suppress these increased noise levels will result in blurring of the image features (Figure 3.6(c)), while the level of residual noises in the background is still significant . These can be compared with the restored image using the current algorithm in Figure 3.6(d), which results in a reasonable appearance for the image even under the current level of severe degradation.

However, we can notice a number of problems with our current approach in Figure 3.6(d). Although most of the background noises are effectively removed compared with the non-adaptive approaches, we can notice some noises around the edges, and some smooth blotches in the textured area due to the classification of some weak textures as smooth regions. These are also present in the 30dB case in Figure 3.5 as well but to a less serious extent. These problems and their solutions will be further discussed in chapters 5 and 6.

It is not sufficient just to employ adaptive processing, but that correct segmentation of the image into smooth and edge/textured regions, and assignment of the appropriate

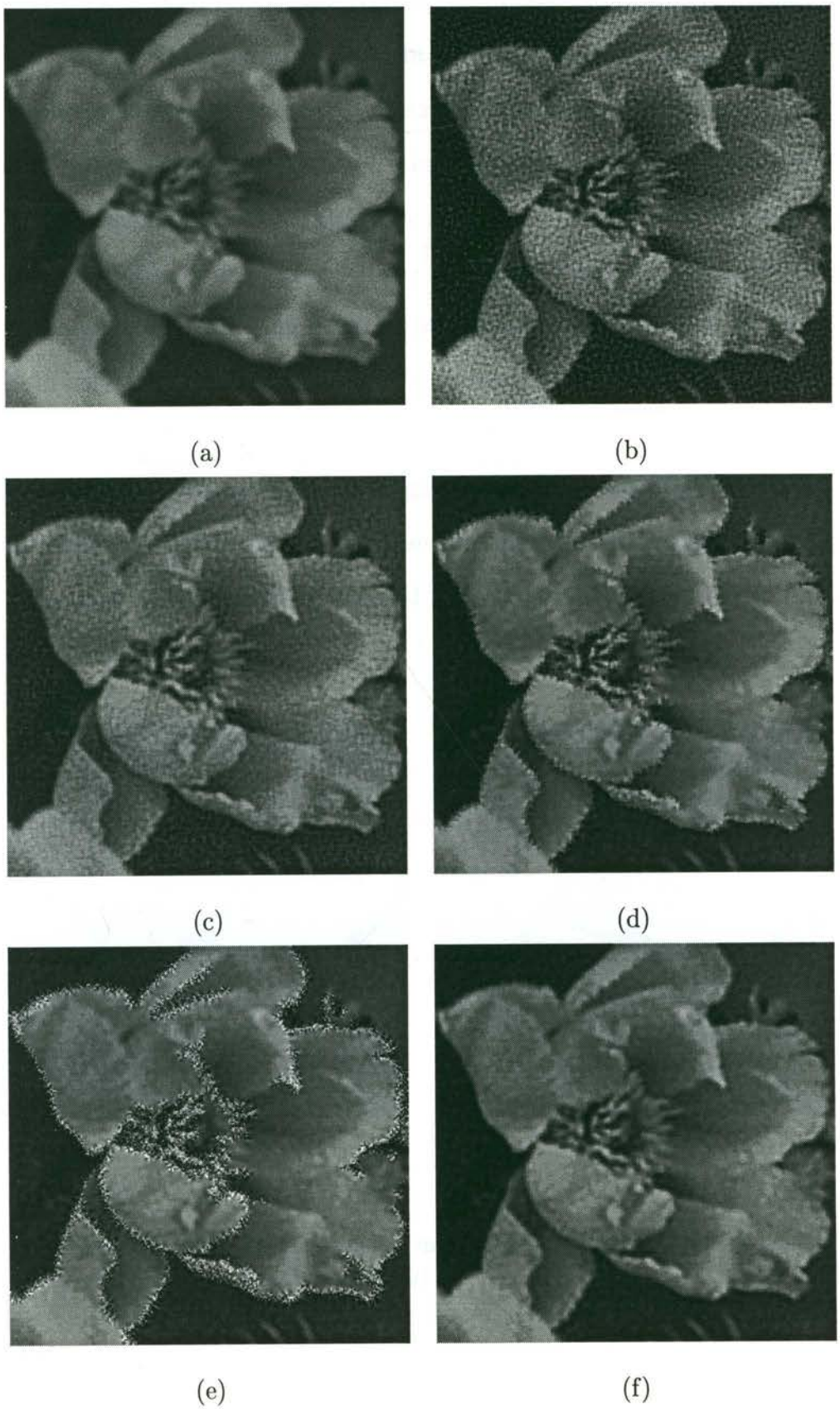
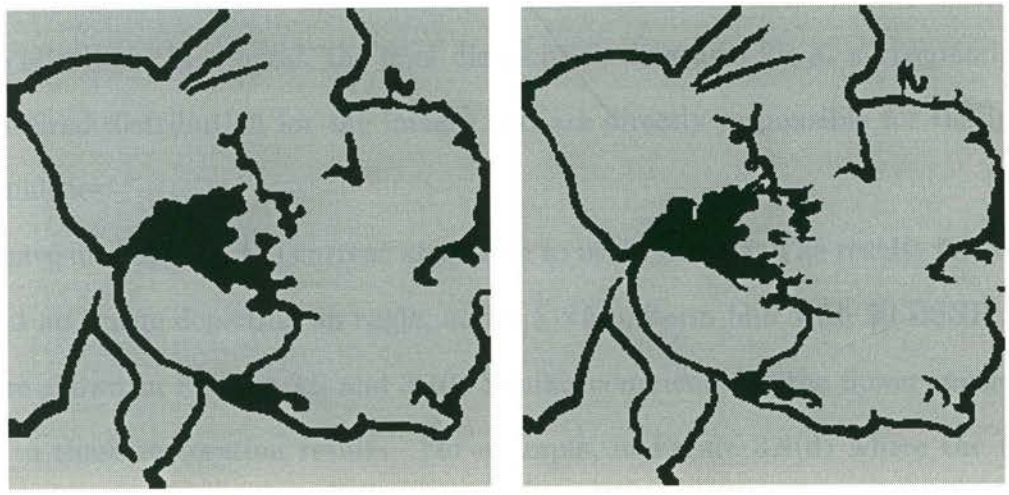


Figure 3.6: Restored Flower images (5×5 uniform blur, 20dB BSNR) (a) Blurred image. (b)-(f) Restored images using (b) non-adaptive Hopfield restoration algorithm (small λ). (c) non-adaptive Hopfield restoration algorithm (optimally adjusted λ by user) (d) HMBNN (e) Decreased λ from HMBNN value in edges/textures (f) Increased λ from HMBNN value in edges/textures.

parameter values to each region, are required to obtain the best visual appearance for the restored image. To illustrate this point, we have included examples of adaptively regularized images which do not satisfy the above conditions. In Figures 3.6(e) and (f), we perturb the regularization parameter values assigned to the combined edge/textured regions by the current algorithm. In Figure 3.6(e), the parameter is adjusted such that it is smaller than the one assigned by the network. We can immediately notice the noisy appearance around the edges and in the textured regions. Although the noises are restricted to only a small area, they severely interfere with the perception of the overall image. On the other hand, if we adjust the parameter in the edge/textured regions such that its value is greater than the originally assigned one (Figure 3.6(f)) , we can notice the blurring of these regions. These illustrate the importance of choosing the correct parameter values, which is achieved by the current algorithm, even if adaptive regularization is adopted.



(a) (b)

Figure 3.7: Evolution of the segmentation map of flower during the restoration process (a) Initial segmentation (b) Final segmentation

In Figure 3.7 , we illustrate the evolution of the boundary of each region, which was continually updated using the nearest neighbor decision criterion (3.64) throughout the restoration process. Starting from the initial segmentation in Figure 3.7(a), the boundary of each region gradually evolves to the final partitions in Figure 3.7(b). Comparing the

partition in Figure 3.7(b) with the initial partition, we can immediately note the closer resemblances of the final partitions to the original images. We can particularly note the more intricate borders in the final partitions resulting from the continual refinement of the boundary pixel classification. In this way, the final segmentation gives a more accurate description of the image contents in a perceptual sense.

The evolution of the λ_r values across the various regions are shown in the images in Figures 3.8(a) and (d). Starting from a uniform λ value across the whole image, the parameter values evolve differently in each region as depicted from Figures 3.8(a) to (d). It can be seen that the regularization parameters decay to small values (corresponding to darker pixels on the diagram) at the textured regions, bringing out the details while increasing towards large values (corresponding to brighter pixels) for smooth regions to achieve noise suppression. Despite the piecewise constant appearance of the diagrams, there exists small local variations of λ_r in each region due to our implementation of the parameter update as a stochastic gradient procedure [41], which serves to adapt to the local peculiarities. In general, the final distribution diagrams for λ_r corresponds closely to our desired distribution for the images and are directly responsible for the improved visual qualities.

We have also applied the current algorithm to other images. The results for the image Lena and an image depicting an eagle, under 5 $\times$5 uniform blur with 30 BSNR additive noise, are shown in Figures 3.9 and 3.10. Similar comments for the flower image can be applied to these restoration results. For example, in Figure 3.9(d) where the adaptive result for the image Lena is shown, we can observe that the smooth regions are correctly regularized compared with the non-adaptive approaches, but the edges are slightly noisy and there are smooth blotches in the textured regions. These problems will be addressed in chapters 5 and 6. Similar comments also apply to the adaptively regularized eagle image in Figure 3.10(d).

Figure 3.11 shows the λ-distribution maps for the three images under 5 $\times$5 uniform blur at different levels of additive noise. Figures 3.11(a) and (b) show the λ-map of the restored flower image under 30dB BSNR and 20dB BSNR respectively, Figures 3.11(c)

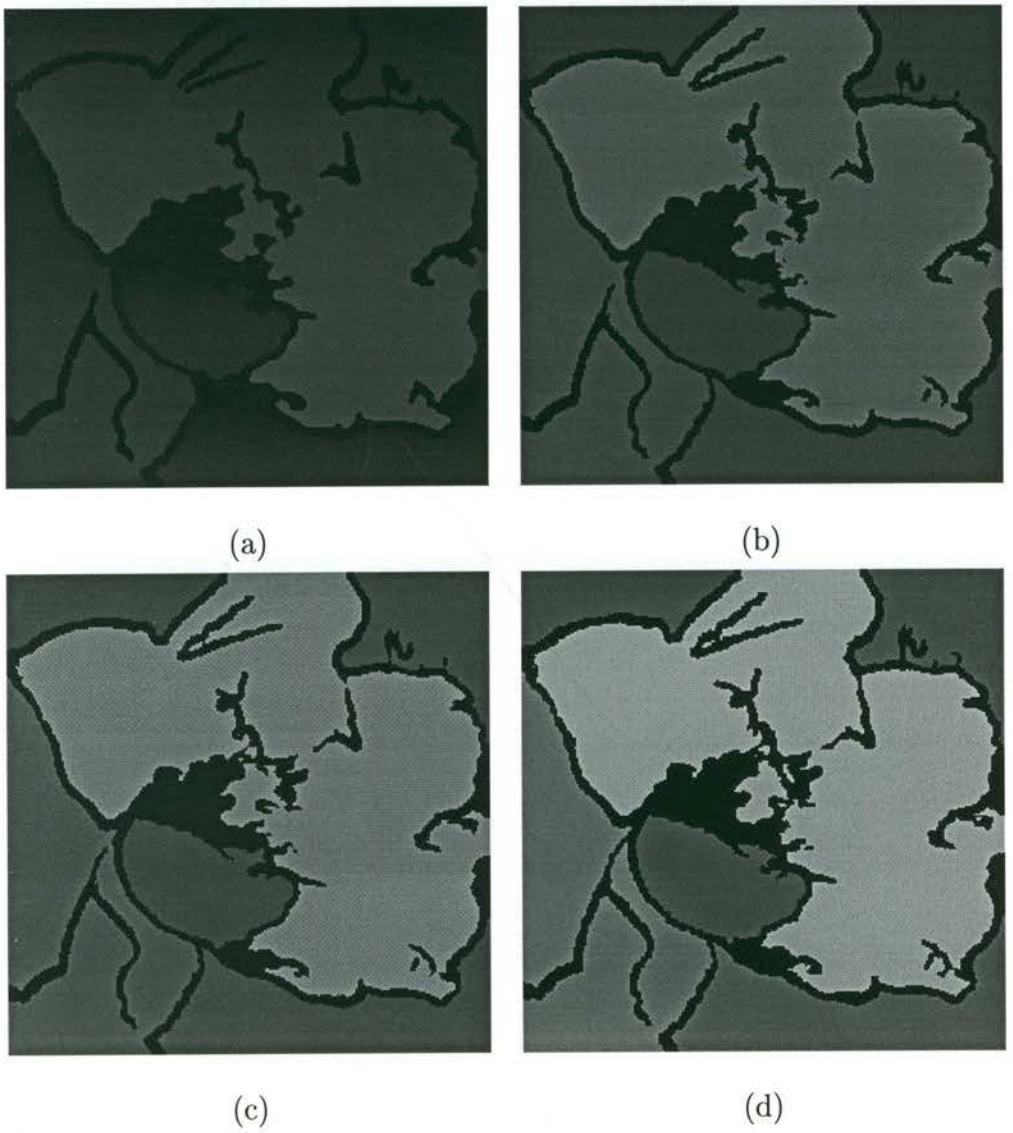


Figure 3.8: Evolution of the regional regularization parameters in the flower image during the restoration process (a)-(d) λ-map (a) iteration 1 (b) iteration 5 (c) iteration 10 (d) final map.

Figure 3.9: Restored Lena images (5×5 uniform blur, 30dB BSNR). (a) Blurred image. (b)-(d) Restored images using (b) non-adaptive Hopfield restoration algorithm (small λ). (c) non-adaptive Hopfield restoration algorithm (optimally adjusted λ by user) (d) HMBNN.

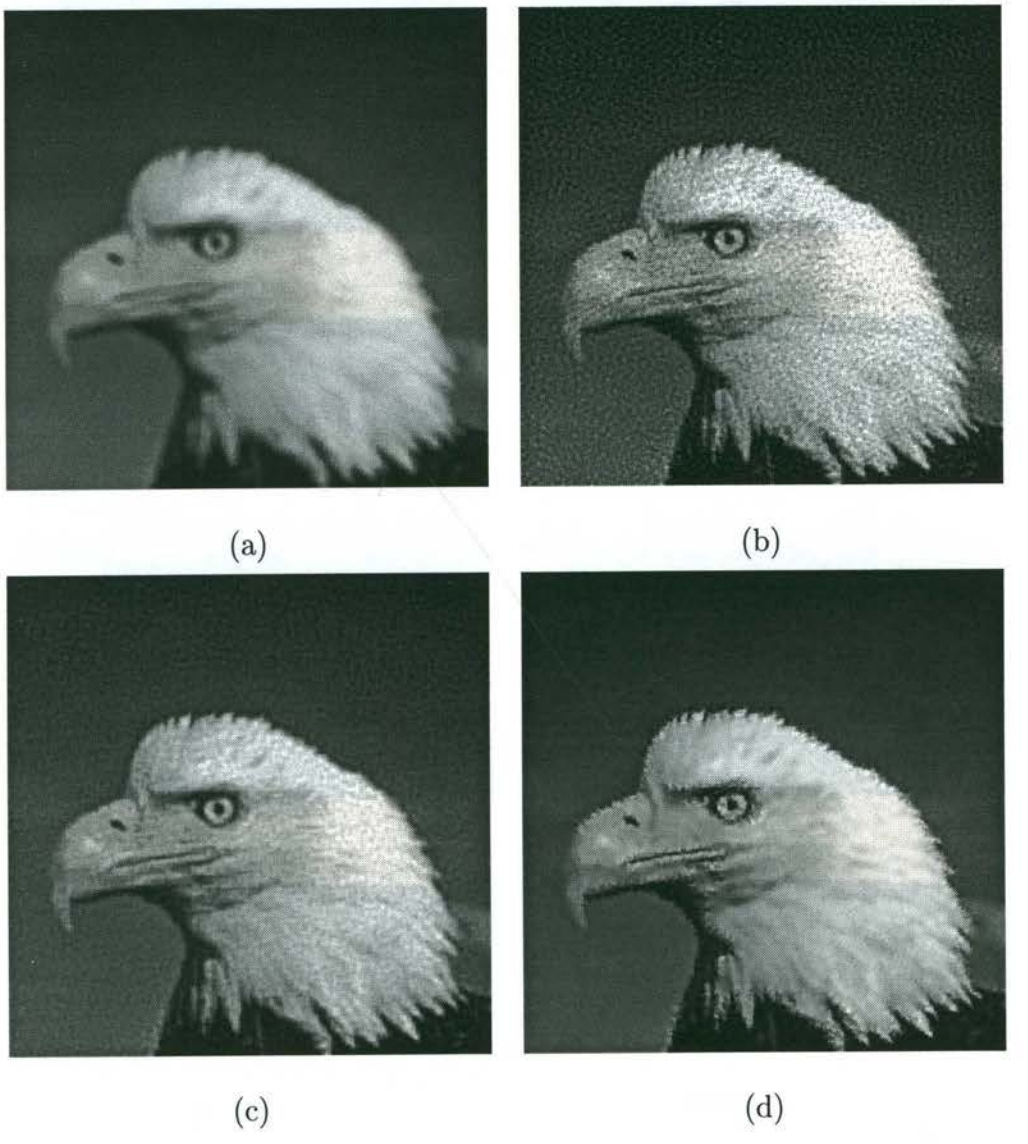


Figure 3.10: Restored Eagle images (5×5 uniform blur, 30dB BSNR). (a) Blurred image. (b)-(d) Restored images using (b) non-adaptive Hopfield restoration algorithm (small λ). (c) non-adaptive Hopfield restoration algorithm (optimally adjusted λ by user) (d) HMBNN.

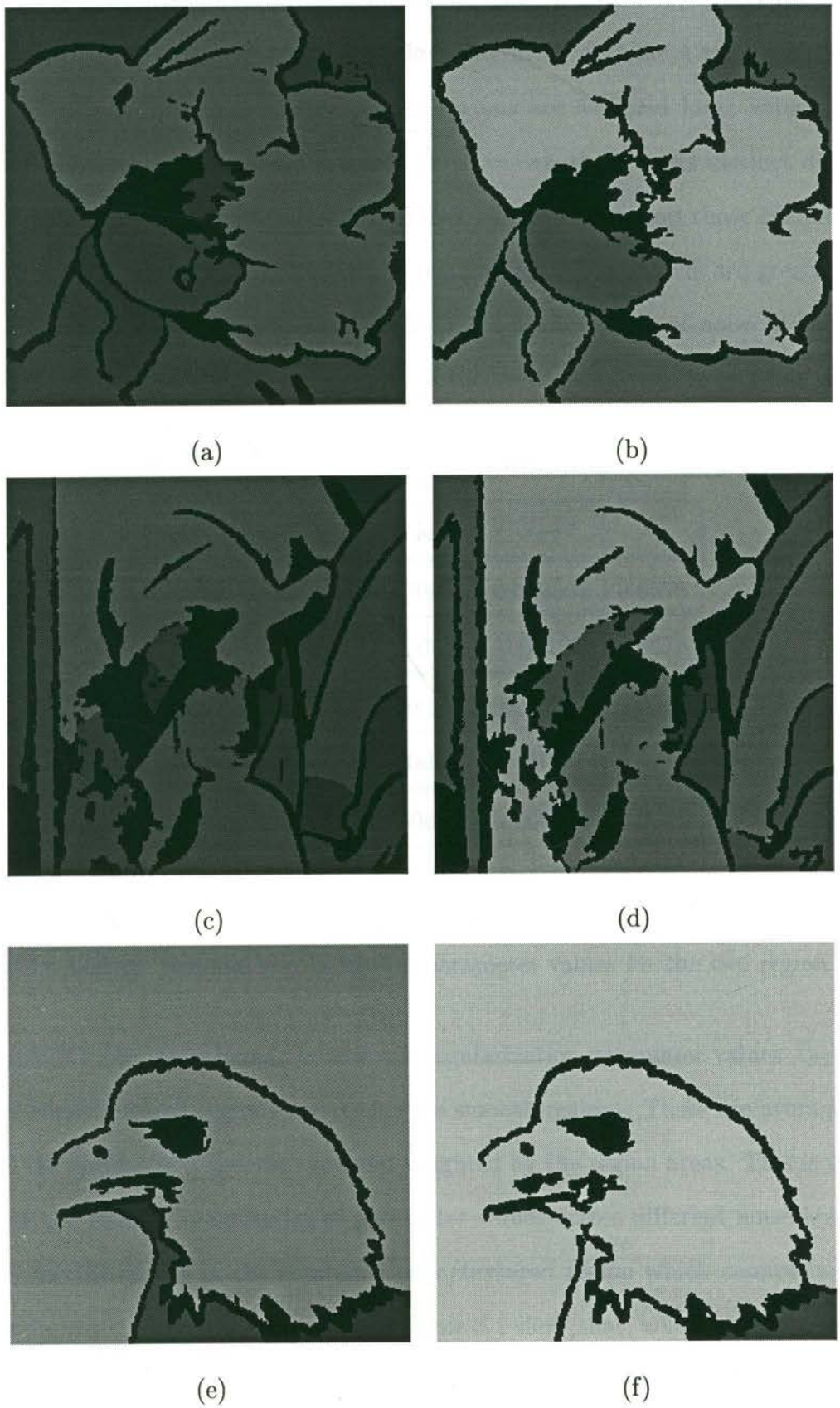


Figure 3.11: The λ-distribution maps for the three images under 5 $\times$5 uniform blur at different levels of additive noise. (a) Flower (30dB BSNR) (b) Flower (20dB BSNR) (c) Lena (30dB BSNR) (d) Lena (20dB BSNR) (e) Eagle (30dB BSNR) (f) Eagle (20dB BSNR).

and (d) show the corresponding maps for Lena, and Figures 3.11(e) and (f) show the corresponding maps for eagle. Apart from the observation that the λ values are correctly assigned in each map such that the smooth regions are assigned large values and the combined edge/texture regions are assigned small values, there exists distinct differences between the maps corresponding to 30dB BSNR additive noise and those corresponding to 20dB BSNR. Generally, the values of the regularization parameters are greater for the 20dB cases than the 30dB cases, which reflects the higher levels of noise in the former cases and the greater necessity for smoothing in the background regions by adopting larger λ values.

Image name,PSF/noise level	$\overline{\lambda_{ET}}$	$\overline{\lambda_S}$
Flower, Uniform PSF/30dB	0.000392	0.0373
Flower, Uniform PSF/20dB	0.00196	0.0678
Lena, Uniform PSF/30dB	0.000349	0.0300
Lena, Uniform PSF/20dB	0.00171	0.0558
Eagle, Uniform PSF/30dB	0.000426	0.0649
Eagle, Uniform PSF/20dB	0.00269	0.0989

Table 3.1: Average regional regularization parameter values for the two region types

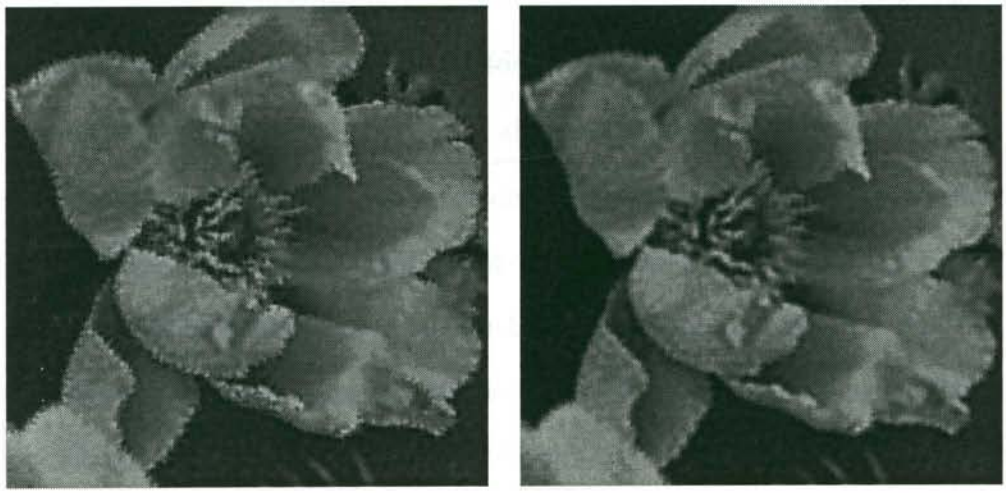
In Table 3.1 , we have listed the average regularization parameter values $\overline{\lambda}_{ET}$ for the combined edge/textured regions and $\overline{\lambda}_S$ for the smooth regions. These are average values across all the regions of a specific type and weighted by the region areas. This is useful in comparing the relative magnitudes of parameter values across different noise levels, and especially for the values in the combined edge/textured region which cannot be readily perceived from the λ- maps. The values in Table 3.1 show that, whenever the level of the additive noise increases, the regularization parameters in different regions are assigned larger values by the algorithm. This is reasonable due to the more urgent need of noise suppression compared with feature enhancements in the more severely degraded images.

(a) (b)

Figure 3.12: Degraded images (5 × 5 uniform blur, 20dB BSNR). (a) Lena (b) Eagle

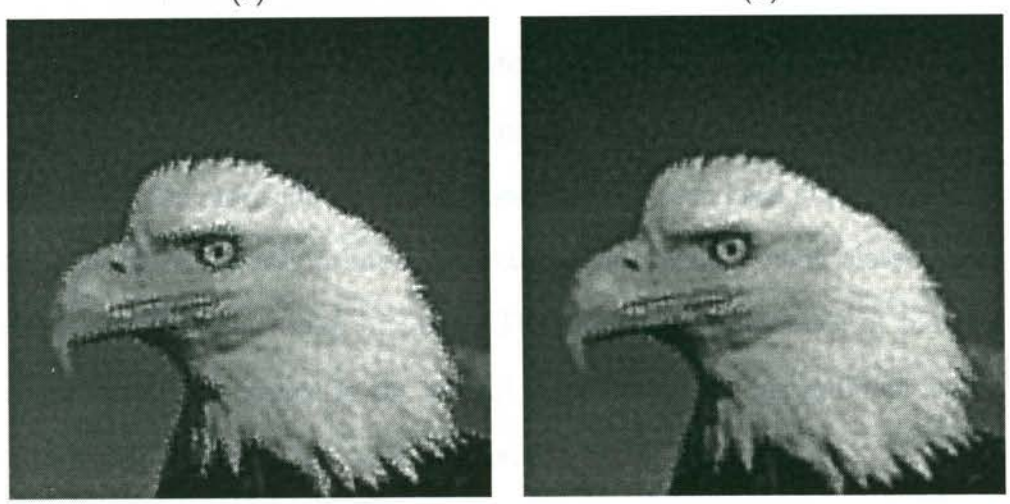
Although the current algorithm gives a reasonable appearance to the restored flower image even for the more severely degraded case, some of the problems become apparent when the current algorithm is applied to other examples of severely degraded images. These can be seen in Figures 3.13(a),(c) and (e), where the current algorithm was applied to the images flower, Lena and eagle under 5 × 5 uniform blur with 20 dB BSNR additive Gaussian noise respectively (The degraded image of flower at 20dB BSNR is shown in Figure 3.6, the degraded images of Lena and eagle at 20dB BSNR are shown in Figures 3.12(a) and (b) respectively). Although the appearance of the restored flower image is reasonable (Figure 3.13(a)) , we can nevertheless perceive slight distortion around the edges of the petals due to the amplified noises. This distortion is more apparent in the Lena image (Figure 3.13(c)) , and is the result of its textured area (feathers on Lena's hat) being connected to some of the edges as a single region as is seen in its λ-map. As a result, essentially the same parameter value is assigned to both types of features. Whereas this particular parameter value is appropriate for the textured area as seen from the image, the same value results in a noisy appearance for the edges. The same problem is also observed in the eagle image (Figure 3.13(e)).

In other words, the current scheme of generating the predicted gray level value using



(a) (b)

(c) (d)



(e) (f)

Figure 3.13: Restored images (5 × 5 uniform blur, 20dB BSNR). (a) Flower (edge-oriented prediction) (b) Flower (texture-oriented prediction) (c) Lena (edge-oriented prediction) (d) Lena (texture-oriented prediction) (e) Eagle (edge-oriented prediction) (f) Eagle (texture-oriented prediction).

$\tilde{v}^{(7)}_{i_1,i_2}$ when $\tilde{v}_{i_1,i_2} \geq \bar{\bar{v}}_{(i_1,i_2)}$, and using $\tilde{v}^{(3)}_{i_1,i_2}$ when $\tilde{v}_{i_1,i_2} < \bar{\bar{v}}_{(i_1,i_2)}$, results in a regularization parameter value which is more appropriate for textures than for edges. To allow increased level of smoothing for the edges, the only alternative choice for generating the desired network output is to adopt the pair of order statistics one step closer to the median, i.e., $\tilde{v}^{(6)}_{i_1,i_2}$ and $\tilde{v}^{(4)}_{i_1,i_2}$. The median itself is not used as the predictor for the edge/textured regions, as its value is usually close to the local mean for blurred or partially restored image, and its adoption will cause over-smoothing of the image features. Therefore, our alternative scheme can be summarized as follows:

$$\tilde{v}^{d}_{i_1,i_2} = \begin{cases} \tilde{v}^{(4)}_{i_1,i_2} & \tilde{v}_{i_1,i_2} < \bar{\bar{v}}_{i_1,i_2} \\ \tilde{v}^{(6)}_{i_1,i_2} & \tilde{v}_{i_1,i_2} \geq \bar{\bar{v}}_{i_1,i_2} \end{cases} \tag{3.66}$$

We will hereafter refer to this modified prediction scheme as the *edge-oriented* prediction scheme, and the original as the *texture-oriented* prediction scheme. As its name implies, we can expect that the resulting regularization parameters under the modified prediction scheme will be more appropriate for edges than for textures, in contrast with the original prediction criterion which is more suitable for textures. The restoration results for this alternative scheme are shown in Figures 3.13(b), (d) and (f). Although the noises in the vicinity of the edges are mostly suppressed in these images, we can notice that the textured areas become over regularized.

In addition to these , we can notice patches of blurred regions among the textured areas in the Lena image. This is due to the need for the algorithm to increase the area of the smooth regions to more effectively suppress the increased level of noises. However, this also leads to parts of the less prominent textured sub-regions being classified as smooth regions, as is seen in the λ-map of Lena. The resulting appearance of blurred patches lends an unnatural appearance to the overall textured area. This problem is also present in the flower image but to a less serious extent.

In view of these problems, it is clearly necessary to derive a new classification procedure which can distinguish between edges and textured areas in an image. We can then apply different regularization strategies to each of these types of features according to their noise masking capabilities. Conventional classification approaches based on the

local image variance or the edge magnitudes usually cannot distinguish these two feature types due to their similar magnitudes of gray level variations. Previous attempts have been made to extract the textured areas in an image [66] , but this requires the evaluation of multiple image attributes and the characterization of particular intervals of each attribute corresponding to textures. In Chapter 5, we have derived a new *scalar* measure, the Edge-Texture Characterization (ETC) measure, which is capable of distinguishing between edges and textured regions using a scalar value alone. In Chapter 6, we apply this measure in a fuzzified version of the current HMBNN which adopts different regularization strategies for edges and textures, thus alleviating the problem previously described. The remaining problem of blurred patches appearance in the textured area clearly requires an alternative characterization of the textured areas *independent* of the local image variances on which the segmentation of the image are based. In Chapter 5, we have derived a textured area extraction algorithm based on the ETC measure which approximately locates the textured regions within the image. This *texture map* can be used as a complementary source of information for the proper regularization of especially those less prominent areas of textures

3.7 Summary

An alternative formulation of the adaptive regularization problem in image restoration was proposed in the form of a *model-based neural network with hierarchical architecture* (HMBNN) operating on non-overlapping regions on the image. Based on the principle of adopting small parameter values for the textured regions for detail emphasis while using large values for ringing and noise suppression in the smooth regions, we have developed an alternative viewpoint of adaptive regularization which centers on the concept of *regularization parameters as model-based neuronal weights.* From this we have derived a stochastic gradient descent algorithm for optimizing the parameter value in each region. In addition, incremental re-definition of the various regions using the nearest neighbor principle is incorporated to reverse the effects of initial inaccurate segmentation.

The current scheme was applied to a number of images containing various combinations of smooth regions, textures and edges. The experimental results correspond closely to our original expectation in that the algorithm automatically adopts small regularization parameters for the detailed regions and large parameter values for the comparatively smooth background.

With the establishment of this new framework of adaptive regularization, various possibilities for further research naturally suggest themselves. For example, we have treated the edge and textured regions as equivalent in this work, but the noise masking capabilities of these two types of regions are in effect very different and is particularly noticeable under high levels of noise. As a result, it would be beneficial to separate these two region types and adopt different regularization strategies for each of them. This is achieved in Chapter 5 where the ETC measure, which is capable of distinguishing between pixel configurations corresponding to edges and textures, is introduced. This is incorporated in Chapter 6 into the present HMBNN framework to result in a fuzzified version of the current network, which is capable of applying different levels of regularization to edges and textures, and thus alleviating many of the problems described in the experimental section. In the meantime, we explore the possibility of applying the current HMBNN framework to the problem of edge characterization and detection in the next Chapter.

Chapter 4

Application of HMBNN to Edge Characterization

4.1 Introduction

In this Chapter, we investigate the feasibility of employing human-defined features as training inputs for neural networks specially designed for feature detection. Specifically, we have investigated the efficiency of a *model-based* neural network with hierarchical architecture [21, 65, 70] in acquiring accurate characterization of what humans regard as features through a Human-Computer Interaction (HCI) approach, and in generalizing this acquired knowledge to novel features which are equally significant but have not been explicitly specified in the training data.

Edge characterization constitutes an important sub-branch of feature extraction where the primary aim is to detect those pixels in an image with specific types of changes in intensities [36, 50] . In view of this, the process is usually divided into two stages: one, the detection of specific types of change; two, a decision about whether they are to be accepted as relevant features to a given problem domain. Typical examples of difference measure used in the first stage include the Roberts operator, the Prewitt operator and the Sobel operator [36, 50]. The second stage is defined by a decision process on the resulting edge magnitudes. This work is particularly concerned with this latter aspect of

edge detection.

In simple edge detection, a global threshold on the edge magnitudes is usually interactively chosen to produce a binary image specifying the edge locations. The result is usually not satisfactory due to the overlapping of the edge magnitude ranges of important features and strong background noise. More sophisticated approaches, including the Canny edge detector [22] and the Shen-Castan edge detector [98], employ an expanded threshold set in the so-called hysteresis thresholding operation, where a significant edge is defined as a connected series of pixels with the edge magnitude of at least one member exceeding an upper threshold, and with the magnitudes of the other members exceeding a lower threshold. In addition, the requirement for exact edge locations usually requires some form of Laplacian of Gaussian (LoG) filtering [74] to detect the zero crossings, or analogous filtering operations in which more parameters have to be specified to determine the proper image resolution at which the edge detection operation is to take place. This, together with the previous thresholding parameters, gives rise to a large variety of possible parameter combinations, each of which will result in a very different appearance for the final edge map.

In this Chapter we have explored a neural computing approach to estimating such parameters to fit human performance. It is our anticipation that still more parameters will be required for future edge detection operations. In view of this, a logical choice for a proper representation of these parameters would be in the form of connection weights for a neural network. To do this, we have developed a model-based neural network for edge characterization which is based on the hierarchical architecture proposed by Kung and Taur [65]. In this case the connection weights of the network encode both the edge-modelling parameters in the high-pass filtering stage and the thresholding parameters in the decision stage. In other words, the input to the network is the original gray level image, and the output is a binary image which specifies the location of the detected edges. This is in contrast with previous uses of NN for edge detection [17, 16, 100] , where the primary motivation is to generalize the traditional differencing operators by replacing them with NN. As a result, the emphasis of their approaches is on the learning of the

filter coefficients in the first filtering stage, and the thresholds for the final decision stage are usually chosen heuristically.

4.2 Network Architecture

The proposed architecture is composed of a hierarchical structure consisting of clusters of neurons forming sub-networks, with each neuron of a sub-network encoding a particular subset of the training samples. In the training stage, each sub-network encodes different aspects of the training set, and in the recognition stage an arbitration process is applied to the outputs of the various sub-networks to produce a final decision.

The motivation for using this HMBNN architecture is that, in edge detection, it would be more natural to adopt multiple sets of thresholding decision parameters and apply the appropriate set of parameters as a function of the local context, instead of just adopting a single set of parameters across the whole image as in traditional approaches.

The HMBNN architecture thus constitutes a natural representation of the above adaptive decision process if we designate each sub-network to represent a different background illumination level, and each unit in the sub-network to represent different prototypes of edge-like features under the corresponding illumination level. The architecture of the feature detection network is shown in Figure 4.1. The internal representation scheme for edge information at the various hierarchical levels are explained below.

4.2.1 Characterization of Edge Information

For illustrative purposes we consider a 3×3 moving window over the image. Representing the gray-level values in the window $\mathcal{W}$ as a vector $\mathbf{x} = [x_1 \quad \ldots \quad x_9]^T \in \mathbf{R}^9$, and defining the mean of these values as follows,

$$\overline{x} = \frac{1}{9}\sum_{n=1}^{9} x_n \tag{4.1}$$

we summarize the gray-level information in the window in terms of a vector $\mathbf{m} = [m_1 \;\; m_2]^T \in \mathbf{R}^2$, which characterizes the two dominant gray level values within the window and is de-

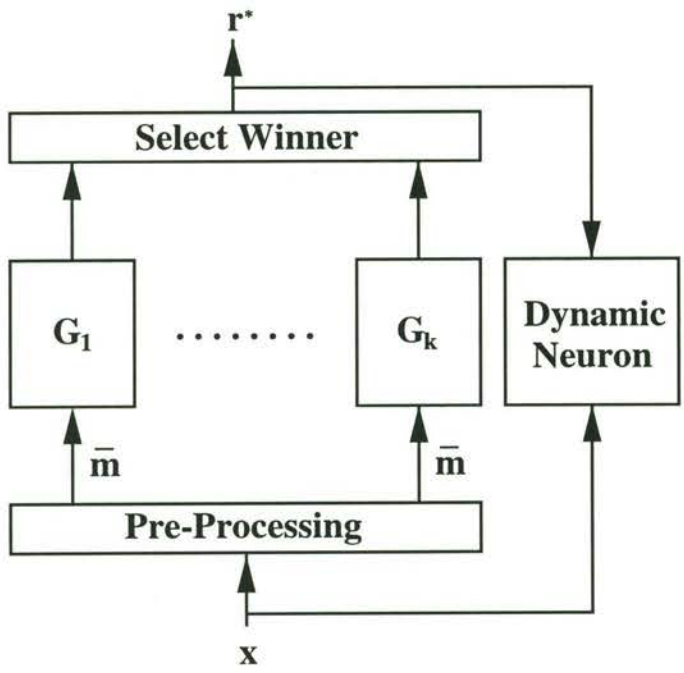


Figure 4.1: The architecture of the HMBNN edge detector

fined as follows

$$\mathcal{M}_1 = \{x_n : x_n < \overline{x}\} \tag{4.2}$$

$$\mathcal{M}_2 = \{x_n : x_n \geq \overline{x}\} \tag{4.3}$$

$$m_1 = \frac{1}{|\mathcal{M}_1|} \sum_{x_n \in \mathcal{M}_1} x_n \tag{4.4}$$

$$m_2 = \frac{1}{|\mathcal{M}_2|} \sum_{x_n \in \mathcal{M}_2} x_n \tag{4.5}$$

$$\overline{m} = \frac{m_1 + m_2}{2} \tag{4.6}$$

4.2.2 Sub-Network G_r

We associate each sub-network $G_r, r = 1, \ldots, R$ with a prototype background illumination gray-level value g_r (the choice of the appropriate number of sub-networks R will be discussed in Section 4.4.3). We then assign all those 3×3 windows in the image with their background gray-level values closest to g_r to the sub-network G_r. Specifically, a particular window $\mathcal{W}$ in the image, with its associated parameters $\overline{m}, m_1, m_2$ defined in Eqs (4.2)

to (4.6) , is assigned to the sub-network G_{r^*} if the following conditions are satisfied

$$g_{r^*} \in [m_1, m_2] \tag{4.7}$$

$$|\overline{m} - g_{r^*}| < |\overline{m} - g_r| \quad r = 1, \ldots, R, r \neq r^* \tag{4.8}$$

where $[m_1, m_2]$ is the closed interval with m_1, m_2 as its endpoints (this is in contrast with the column vector representation of $\mathbf{m}$, which we denote as $[m_1 \;\; m_2]^T$). In this way, we partition the set of all 3×3 windows in the image into clusters, with all members in a single cluster exhibiting similar levels of background illumination. Hereafter we denote the conditions in (4.7) and (4.8) collectively as $\mathbf{x} \longrightarrow G_{r^*}$. The operation of this sub-network assignment process is illustrated in Figure 4.1.

4.2.3 Neuron N_{rs} in Sub-Network G_r

Each sub-network G_r contains S neurons $N_{rs}, s = 1, \ldots, S$, with each neuron encoding the various different edge prototypes which can exist under the general illumination level g_r. We associate each neuron with a weight vector $\mathbf{w}_{rs} = [w_{rs,1} \;\; w_{rs,2}]^T \in \mathbf{R}^2$, which serves as a prototype for the vector $\mathbf{m}$ characterizing the two dominant gray-level values in each 3×3 window $\mathcal{W}$. We assign a certain window with corresponding vector $\mathbf{m}$ to the neuron N_{rs^*} if the following condition is satisfied

$$\|\mathbf{m} - \mathbf{w}_{rs^*}\| < \|\mathbf{m} - \mathbf{w}_{rs}\| \quad s = 1, \ldots, S, s \neq s^* \tag{4.9}$$

In this work, we have chosen $S = 2$ in order that one of the neurons encodes the prototype for weak edges and the other encodes the prototype for strong edges. We designate one of the weight vectors $\mathbf{w}_{rs}, s = 1, 2$ as the weak edge prototype $\mathbf{w}_r^l$ and the other one as the strong edge prototype $\mathbf{w}_r^u$ according to the following criteria:

- $\mathbf{w}_r^l = \mathbf{w}_{rs_l}$, where $s_l = \arg\min_s(w_{rs,2} - w_{rs,1})$
- $\mathbf{w}_r^u = \mathbf{w}_{rs_u}$, where $s_u = \arg\max_s(w_{rs,2} - w_{rs,1})$

Given the weak edge prototype $\mathbf{w}_r^l = [w_{r,1}^l \;\; w_{r,2}^l]^T$, the measure $(w_{r,2}^l - w_{r,1}^l)$ plays a similar role as the threshold parameter in conventional edge detection algorithms in specifying the

lower limit of visibility for edges, and is useful for identifying potential starting points in the image for edge tracing. The operation of the neuron assignment process is illustrated in Figure 4.2.

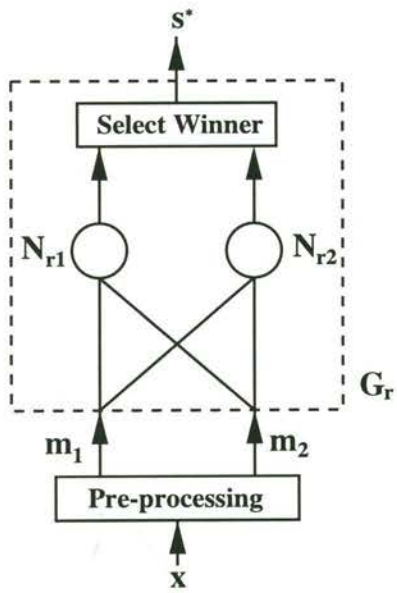


Figure 4.2: The architecture of a single sub-network G_r

4.2.4 Dynamic tracking neuron N_d

Independent of the neuron clusters G_r and N_{rs}, we associated a *dynamic tracking neuron* N_d with the network. This neuron is global in nature in that it does not belong to any of the sub-networks G_r, and that all of the sub-networks can interact with this particular neuron. Similar to the neurons N_{rs} in each sub-network G_r, the dynamic neuron consists of a *dynamic weight vector* $\mathbf{w}_d = [w_{d,1} \quad w_{d,2}]^T \in \mathbf{R}^2$. In addition, it consists of a scalar parameter g_d which is analogous to the illumination level indicator g_r of each sub-network. Thus this particular neuron takes on both the characteristics of a sub-network and a neuron. The structure of the dynamic tracking neuron is shown in Figure 4.3.

The primary purpose of this neuron is to accommodate the continuously varying characteristics of an edge as it is being traced out in the *recognition* phase. As opposed to the usual practice of varying the neuronal weights during the *training* phase, this neuron is inactive during the acquisition of the edge configurations. It is only during the recog-

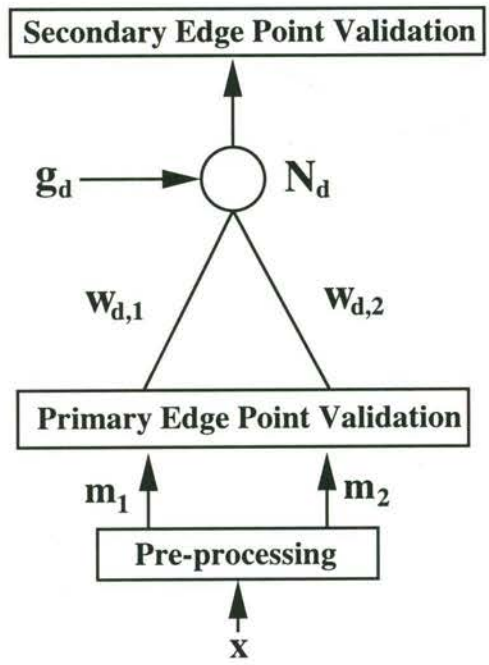


Figure 4.3: The structure of the dynamic edge tracking neuron

nition phase and upon the detection of a primary edge point, which is a comparatively prominent image feature location (the exact definition is given in a later section), that we dynamically vary the weight vector $\mathbf{w}_d$ and illumination level indicator g_d of this neuron to trace out all those secondary edge points connected to the primary edge point. The detailed operation of this neuron is described in a later section.

4.2.5 Binary edge configuration

Suppose that the vector $\mathbf{m}$ for the current window $\mathcal{W}$ is assigned to neuron N_{rs} with weight vector $\mathbf{w}_{rs}$. We can define the function $\mathcal{Q} : \mathbf{R}^9 \times \mathbf{R}^2 \longrightarrow \mathbf{B}^9$,where $\mathbf{B} = \{0, 1\}$, which maps the real vector $\mathbf{x} \in \mathbf{R}^9$ representing the gray-level values of the current window to a binary vector $\mathbf{b} = \mathcal{Q}(\mathbf{x}, \mathbf{w}_{rs}) = [q(x_1, \mathbf{w}_{rs}) \quad \ldots \quad q(x_9, \mathbf{w}_{rs})]^T \in \mathbf{B}^9$, in terms of the component mapping $q : \mathbf{R} \times \mathbf{R}^2 \longrightarrow \mathbf{B}$ as follows:

$$q(x_n, \mathbf{w}_{rs}) = \begin{cases} 0 & \text{if } |x_n - w_{rs,1}| < |x_n - w_{rs,2}| \\ 1 & \text{if } |x_n - w_{rs,1}| \geq |x_n - w_{rs,2}| \end{cases} \tag{4.10}$$

The binary vector $\mathbf{b}$ assumes a special form for valid edge configurations. Some of the possible valid edge configurations are shown in Figure 4.4.

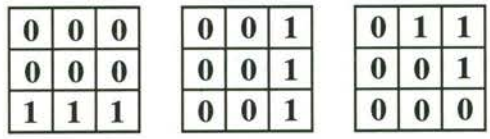


Figure 4.4: Examples of valid edge configurations

During the network training phase , we acquire all the human-specified valid edge configuration patterns in a process to be described in a later section. We will store all these patterns in an *edge configuration set* C which forms part of the overall network. A distinguishing feature of the set C is that it is closed under an operation $R_{\frac{\pi}{4}}$, which permutes the entries of a vector $\mathbf{b} \in \mathbf{B}^9$ in such a way that, when interpreted as the entries in a 3×3 window, $R_{\frac{\pi}{4}}(\mathbf{b})$ is the 45^o clockwise rotated version of $\mathbf{b}$. This operation is illustrated in Figure 4.5. The purpose of imposing this structure on C is that the edge configuration in the training image may not exhibit all of its possible rotated versions, so that the above constraint can facilitate the detection of those rotated edge configurations not present in the training set.

Figure 4.5: Illustrating the operation $R_{\frac{\pi}{4}}$

4.2.6 Correspondence with the General HMBNN architecture

The current edge characterization network has a direct correspondence with the general architecture described in Chapter 2 as follows: the sub-networks $G_r, r = 1, \ldots, R$ can be directly associated with the sub-networks in the general model, and the neurons $N_{rs_r}, s_r = 1, \ldots, S_r$ are associated with those within each sub-network in the general model, with $S_r = 2$ for all r in the current application. Due to our implementation of competitive learning at both the sub-network and the neuronal levels, it belongs to the class of subcluster hierarchical networks as described in Chapter 2.

Instead of embedding a low-dimensional weight vector within a high-dimensional space as in the network in Chapter 3, we have instead mapped the comparatively high-dimensional input vector $\mathbf{x}$ into a low-dimensional input $\mathbf{m}$ which characterizes the two dominant gray level values within a local window of pixels. In other words, we adopt the input-parameterized model-based neuron in Chapter 2 with the mapping $\mathcal{P} : \mathbf{R}^9 \longrightarrow \mathbf{R}^2$, defined such that,

$$\mathbf{m} = \mathcal{P}(\mathbf{x}) \tag{4.11}$$

Despite the above correspondences, there are slight differences between the original formulation of the subcluster hierarchical structure in [65] and the current edge characterization network structure. In the original network, the sub-network output $\phi(\mathbf{x}, \mathbf{w_r})$ is directly substituted by the neuron output of the local winner as follows

$$\phi(\mathbf{x}, \mathbf{w}_r) \equiv \psi_r(\mathbf{x}, \mathbf{w}_{s^*}) \tag{4.12}$$

where s^* is the index of the local winner, and $\psi_r(\mathbf{x}, \mathbf{w}_{s^*})$ is the local neuron output of the winning neuron.

In the current network, on the other hand, the competition process at the sub-network level is independent of the corresponding process at the neuronal level, and is of a very different nature: the competition between the sub-networks is specified in terms of the conformance of the current local illumination gray-level value with one of the prototype background illumination levels, i.e., a comparison between scalar values. The competition at the neuronal level, however, involves the comparison of vectors in the form of two-dimensional edge prototypes existing under a particular illumination gray-level value. As a result, the sub-network outputs and the local neuron outputs are independently specified at the two hierarchical levels. At the sub-network level, the following sub-network output is defined for the background illumination gray level values:

$$\phi(\overline{m}, g_r) = |\overline{m} - g_r| \tag{4.13}$$

At the neuron level, the following local neuron output is defined for the edge prototype vectors:

$$\psi_r(\mathbf{m}, \mathbf{w}_{rs}) = \|\mathbf{m} - \mathbf{w}_{rs}\| \tag{4.14}$$

4.3 Network Training

The training of the network proceeds in three stages: in the first pass, we determine the prototype illumination level $g_r, r = 1, \ldots, R$ for each sub-network G_r by competitive learning [41, 60]. In the second stage, we assign each window $\mathcal{W}$ to its corresponding sub-network and then determine the weight vectors $\mathbf{w}_{rs}$, again using competitive learning. In the third stage, we assign each window to its corresponding neuron in the correct sub-network, and extract the corresponding binary edge configuration pattern $\mathbf{b}$ as a function of the winning weight vector $\mathbf{w}_{rs}$. We apply the operation $R_{\frac{\pi}{4}}$ successively to $\mathbf{b}$ to obtain the 8 rotated versions of this pattern, and insert these patterns into the edge configuration memory C.

4.3.1 Determination of g_r for sub-network G_r

Assuming that the current window with associated parameter $\overline{m}$ is assigned to the sub-network G_r according to the conditions in (4.7) and (4.8) . Using competitive learning, we update the value of g_r using the following equation:

$$g_r(t+1) = g_r(t) + \eta(t)(\overline{m} - g_r(t)) \tag{4.15}$$

The learning stepsize $\eta(t)$ is successively decreased according to the following linear schedule

$$\eta(t+1) = \eta(0)(1 - \frac{t}{t_f}) \tag{4.16}$$

where t_f is the total number of iterations.

4.3.2 Determination of $\mathbf{w}_{rs}$ for neuron N_{rs}

In the second stage, we assume that the current window with associated feature vector $\mathbf{m}$ has been assigned to the neuron N_{rs} within the sub-network G_r. Again, employing competitive learning, we can update the current weight vector $\mathbf{w}_{rs}$ using the following equation

$$\mathbf{w}_{rs}(t+1) = \mathbf{w}_{rs}(t) + \eta(t)(\mathbf{m} - \mathbf{w}_{rs}(t)) \tag{4.17}$$

where the stepsize $\eta(t)$ is successively decreased according to Eq (4.16).

4.3.3 Acquisition of valid edge configurations

In the third stage, after determining the background illumination levels g_r for the sub-networks and the weight vectors $\mathbf{w}_{rs}$ for all the neurons, we once again assign all the 3×3 windows to their corresponding sub-networks and the correct neurons within the sub-networks. Assuming that the current window is assigned to neuron N_{rs} within sub-network G_r. We can transform the current window $\mathcal{W}$ with gray-level vector $\mathbf{x} \in \mathbf{R}^9$ into a binary edge configuration vector $\mathbf{b} \in \mathbf{B}^9$ according to equation (4.10):

$$\mathbf{b} = \mathcal{Q}(\mathbf{x}, \mathbf{w}_{rs}) \tag{4.18}$$

where $\mathbf{w}_{rs}$ is the associated weight vector of N_{rs}. The binary vector $\mathbf{b}$ is thus a *human-specified valid edge configuration* and is inserted into the valid edge configuration set C. To satisfy the requirement that the set C be closed under the operation $R_{\frac{\pi}{4}}$, we generate eight edge configurations $\mathbf{b}_j, j = 0, \ldots, 7$ using $R_{\frac{\pi}{4}}$ as follows

$$\mathbf{b}_0 = \mathbf{b} \tag{4.19}$$

$$\mathbf{b}_{j+1} = R_{\frac{\pi}{4}}(\mathbf{b}_j) \quad j = 0, \ldots, 6 \tag{4.20}$$

and insert all eight patterns into the configuration set C.

4.4 Recognition Phase

In the recognition phase, the network is required to locate all those pixels in a test image with similar properties as the acquired feature prototypes - thus testing the generalization capability of the network. This recognition phase proceeds in two stages. In the first stage, all pixels in the test image with high degree of conformance with the acquired edge prototypes are designated as primary edge points. This is analogous to the stage of hysteresis thresholding in conventional edge detection algorithms where we locate those edge points with their edge magnitudes greater than an upper threshold. In the second

stage, starting from the high conformance edge points, we apply a recursive edge tracing algorithm to locate all those pixels connected to these edge points satisfying a relaxed conformance criterion, which are the secondary edge points. This is analogous to the hysteresis thresholding operation where we specify a lower threshold for all those edge points connected to those points with edge magnitudes above the upper threshold.

4.4.1 Location of primary edge points

In this first stage, we inspect all 3×3 windows in the test image and designate all those windows with corresponding parameters $\overline{m}, m_1, m_2$ as primary edge points, if the following conditions are satisfied

(p1). $\mathbf{x} \longrightarrow G_r$ (refer to conditions (4.7) and (4.8)) for some r.

(p2). $m_2 - m_1 \geq w^l_{r,2} - w^l_{r,1}$, where $\mathbf{w}^l_r$ is the weak edge prototype vector of G_r.

(p3). $\mathbf{b} = \mathcal{Q}(\mathbf{x}, \mathbf{w}_{rs}) \in C$, where $\mathbf{w}_{rs}$ is the weight vector associated with the assigned neuron for $\mathbf{x}$.

Condition (p1) specifies that the background illumination level value of the current window should match one of those levels associated with a sub-network. Condition (p2) ensures that the magnitude of gray-level variation, which is characterized by the difference $m_2 - m_1$, is greater than the corresponding quantity of the weak edge prototype of the assigned subnetwork. Condition (p3) ensures that the binary edge configuration corresponding to the current window is included in the edge configuration memory, C, of the network.

4.4.2 Location of secondary edge points

At this stage, we utilize the dynamic tracking neuron N_d associated with the network to trace the secondary edge points connected to a primary edge point. The weights of the tracking neuron are initialized using the relevant parameters m^p_1, m^p_2 and $\overline{m}^p$ of the

current window $\mathcal{W}$ containing the detected primary edge point.

$$w_{d,1}(0) = m_1^p \tag{4.21}$$

$$w_{d,2}(0) = m_2^p \tag{4.22}$$

$$g_d(0) = \overline{m}^p \tag{4.23}$$

where $\mathbf{w}_d = [w_{d,1} \quad w_{d,2}]^T$ is the weight vector of the dynamic neuron and g_d is a local illumination level indicator which keeps track of the varying gray-level values as the current edge is being traced. As mentioned above, both of them are *dynamic* quantities such that their values are continually updated during the *recognition* phase rather than the *training* phase. We have adopted this approach since during the tracing of an edge, although the primary edge point and its associated secondary edge points may reasonably conform to a particular edge prototype in the trained network, there will inevitably be deviations, especially when the network is applied to a test image which it has not been trained on before. In these cases, the weights of the dynamic neuron will serve as a *specialized* edge prototype as distinguished from the *general* edge prototypes represented by the sub-networks and neurons of the original network. This is especially important for the accurate conversion of the gray level values $\mathbf{x}$ of the current window $\mathcal{W}$ into a binary edge configuration $\mathbf{b}$ to be matched against the valid configurations in the edge pattern memory C.

After the proper initialization of the dynamic neuron, we apply a recursive edge tracing algorithm to locate all those pixels connected to the primary edge points by recursively checking a specific set of *secondary edge validation criteria* at each 8-neighbor of the current primary edge point, and designating those satisfying the criteria as secondary edge points. For a particular primary edge point, we will locate all of its associated secondary edge points using the dynamic neuron. The following set of conditions are specified for the validation of the secondary edge points

(s1). $\mathbf{b} = \mathcal{Q}(\mathbf{x}, \mathbf{w}_d) \in C$, where $\mathbf{w}_d$ is the weight vector associated with the dynamic neuron.

(s2). $g_d \in [m_1, m_2]$, where g_d is the local illumination level indicator associated with the dynamic neuron.

Condition (s1) is similar to the case for primary edge point detection, where we match the corresponding binary edge configuration of the current window with those in the edge configuration memory C. Condition (s2) is a modified version of condition (4.7) in the requirements for $\mathbf{x} \longrightarrow G_r$, which ensures that a potential secondary edge point should be under similar levels of background illumination levels (as summarized by the parameters of the dynamic neuron) as those points in the portion of edge already traced out. Notice that we have allowed the possibility of weak secondary edge points by omitting the constraints on edge magnitude (Condition (p2) for primary edge detection), as long as the points are connected to a primary edge point.

For each validated secondary edge point, we update the local illumination level indicator g_d of the dynamic neuron using the mean value $\overline{m}^s$ of the corresponding window $\mathcal{W}$

$$g_d(t+1) = g_d(t) + \eta(0)(\overline{m}^s - g_d(t)). \tag{4.24}$$

In addition, if the secondary edge point satisfies condition (p2) for *primary* edge point detection - implying that the edge strength of the current point is comparable with that of a primary edge point, we utilize the vector $\mathbf{m}^s$ of the corresponding window to update the weight vector of the dynamic neuron

$$\mathbf{w}_d(t+1) = \mathbf{w}_d(t) + \eta(t)(\mathbf{m}^s - \mathbf{w}_d(t)) \tag{4.25}$$

The purpose of using validated secondary edge points to update g_d is to allow the dynamic neuron to closely track the local gray level values in order for condition (s2) (for secondary edge point detection) to be accurately detected. The decision to use a *fixed* learning step size $\eta(0)$ as opposed to the usual decreasing step size sequence also reflects this purpose. That is, a fixed stepsize allows the gradual decaying of the initial conditions during edge tracing such that the value of g_d will reflect the most current average gray level values in the local neighborhood. On the other hand, the decision to use only those

validated points which satisfy the edge magnitude criterion (p2) for updating $\mathbf{w}_d$ is that, during the edge tracing process, we may encounter very weak edge points which are, nevertheless, validated through their satisfaction of the secondary edge point detection criteria (s1) and (s2). However, the vector $\mathbf{m}^s$ extracted from the current window is unlikely to be representative of the characteristics of the current edge being traced, and thus should not be included in the updating of the local edge prototype encoded in the dynamic weight vector $\mathbf{w}_d$. The learning stepsize $\eta(t)$ in this case is thus allowed to decrease gradually such that the weight vector still partially retains the characteristics of the initial primary edge point from which we start the tracing.

4.4.3 Determination of the network size

We recall that each sub-network G_r is characterized by a weak edge prototype vector $\mathbf{w}_r^l$, and a strong edge prototype vector $\mathbf{w}_r^u$. The corresponding component differences $w_{r,2}^l - w_{r,1}^l$ and $w_{r,2}^u - w_{r,1}^u$ thus reflect the range of edge magnitudes covered by the current sub-network. We define the following *average prototype vector* $\overline{\mathbf{w}}_r = [\overline{w}_{r,1} \;\; \overline{w}_{r,2}]^T \in \mathbf{R}^2$ as follows

$$\overline{\mathbf{w}}_r \equiv \frac{\mathbf{w}_r^l + \mathbf{w}_r^u}{2} \tag{4.26}$$

The interval $[\overline{w}_{r,1}, \overline{w}_{r,2}]$ (as opposed to the vector notation $[\overline{w}_{r,1} \;\; \overline{w}_{r,2}]^T$) thus represents the average gray level range covered by the current sub-network. For a parsimonious representation of the edges, we usually avoid excessive overlapping of these associated intervals of the sub-networks. According to the weight update equation (4.17), if we initialize $\mathbf{w}_{rs}(0)$ such that $g_r \in [w_{rs,1}(0), w_{rs,2}(0)], s = 1, 2$ and if we update $\mathbf{w}_{rs}(t)$ with those vectors $\mathbf{m}$ such that $g_r \in [m_1, m_2]$ as specified in criterion $\mathbf{x} \longrightarrow G_r$, then it is obvious that $g_r \in [w_{rs,1}, w_{rs,2}], s = 1, 2$ for the final weight vector $\mathbf{w}_{rs}$, and therefore $g_r \in [\overline{w}_{r,1}, \overline{w}_{r,2}]$ for the average prototype vector $\overline{\mathbf{w}}_r$. In view of this, we have defined the notion of substantial overlap between intervals of the sub-networks as follows: for sub-network G_r, if there exists one or more sub-networks $G_{r'}$ with $r' \neq r$, such that the condition $g_r \in [\overline{w}_{r',1}, \overline{w}_{r',2}]$ is satisfied as well, then we say that there is substantial

overlapping between the associated intervals of sub-networks G_r and $G_{r'}$.

Adopting the usual definition of the membership function for set A,

$$I_A(a) = \begin{cases} 1 & \text{if } a \in A \\ 0 & \text{if } a \notin A \end{cases} \tag{4.27}$$

we define the *interval overlap measure* O_r^R for sub-network G_r as follows:

$$O_r^R = \sum_{r'=1}^{R} I_{[\overline{w}_{r',1}, \overline{w}_{r',2}]}(g_r) \tag{4.28}$$

which essentially counts the number of sub-networks substantially overlapping with sub-network r (including itself). For example, apart from satisfying the condition $g_r \in [\overline{w}_{r,1}, \overline{w}_{r,2}]$, if there is one other sub-network $r', r' \neq r$ such that the condition $g_r \in [\overline{w}_{r',1}, \overline{w}_{r',2}]$ is satisfied as well, then $O_r^R = 2$.

The *mean interval overlap measure* $\overline{O}^R$ is then defined as follows:

$$\overline{O}^R = \frac{1}{R} \sum_{r=1}^{R} O_r^R \tag{4.29}$$

In general, the values of these two measures depend on the number of sub-networks R in the network.

According to these definitions, a natural criterion to adopt for the choice of R to avoid substantial overlapping between sub-networks, while ensuring adequate representation of the edges in the training images, is to select the maximum possible number of sub-networks R while subjecting to the constraint $\overline{O}^R < 2$ to avoid substantial interval overlapping. More precisely, we select R^* such that

$$R^* = \max\{R : \overline{O}^R < 2\} \tag{4.30}$$

4.5 Experimental Results

For network training, we have selected various edge examples from two images, a flower and an eagle, and incorporated them into the training set. To increase the size of the training set, we have prepared multiple edge tracings by a single user for each of the two images. The eagle image is shown in Figure 4.6(a) and the four independent tracings by

a single user for this image are shown in Figures 4.6(b) to (e). The corresponding image and tracings for the flower image are shown in Figures 4.7(a) to (e). In all cases the task was to "trace out" significant edges in the image.

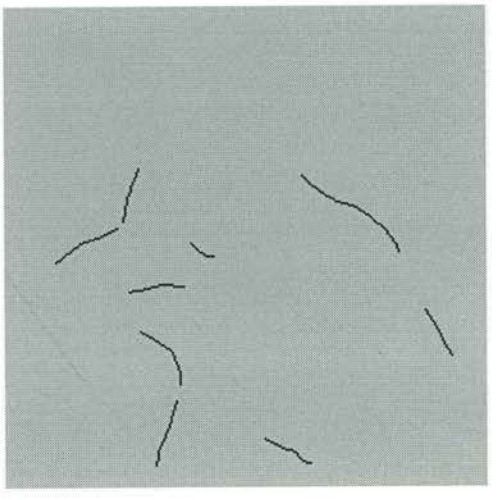
From Figures 4.6 and 4.7, it is evident that the preference of a single user is highly correlated: most of the preferred edge points are re-selected in all the four tracings, although there are variations between trials. Although it is not immediately obvious how these highly correlated tracings can increase the training set size, this can be explained by referring to Figure 4.8. In the figure, the solid curve represents the "true" edge and the dashed curves represent the various different tracings in independent trials. In most of the cases, we only expect the human user to trace the edge approximately, and so each of the independent trials produces tracings which intersect different portions of the true edge. As a result, each independent trial increases the number of training examples corresponding to locations on the true edge.

One may argue that the multiple tracings also increase the number of non-edge points close to the "true" edge due to the inexact tracings, but the very nature of the current network in acquiring new edge configurations will ensure that most of these non-edge points will be automatically rejected: in order for a training edge point with its associated neighboring gray levels $\mathbf{x}$ to modify any weight vector under a certain sub-network G_r, it has to satisfy the condition $\mathbf{x} \longrightarrow G_r$, which requires the checking of the membership condition $g_r \in [m_1, m_2]$. For a non-edge point, the length of the interval $[m_1, m_2]$ will be negligibly small, and it is highly improbable that any gray level indicator g_r associated with the sub-networks will be included in that interval, thus preventing this point from modifying any weight vector in the network.

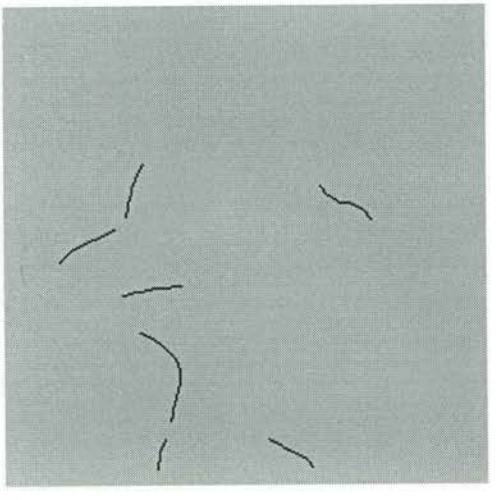
We first applied the NN-based edge characterization scheme to the images employed in the training set. This was done to test the performance of the network in generalizing the sparse tracings of the human users to identify the other edge-like features on the same image. We have applied condition (4.30) in selecting the number of sub-networks, which results in $R^* = 5$. The detected edges for the eagle image using the current method are shown in Figure 4.9(a).

(a)

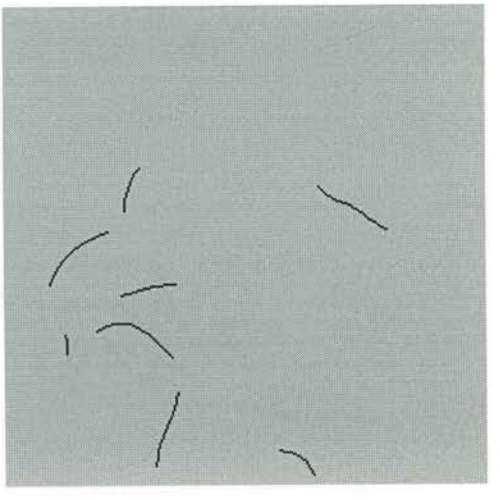
(b)



(c)



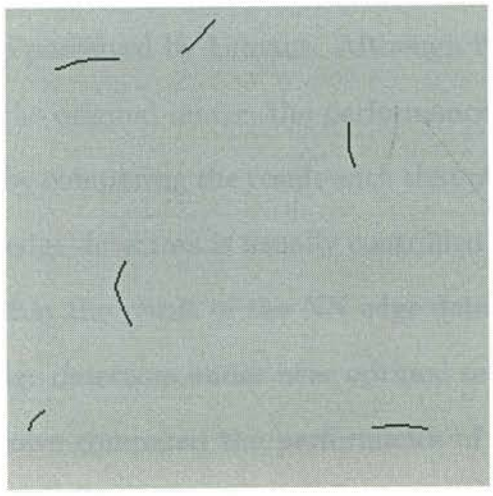
(d)

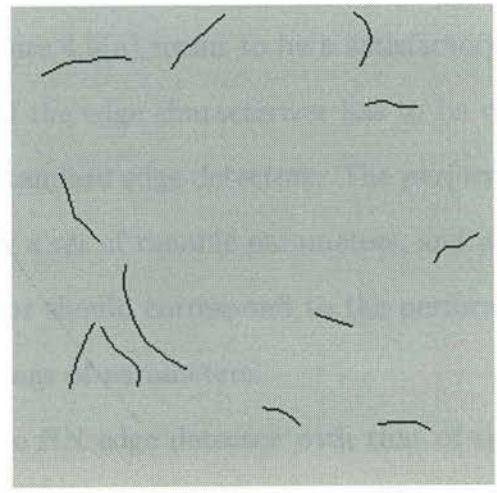


(e)

Figure 4.6: (a) Eagle image (b)-(e) Edge examples supplied by human user.

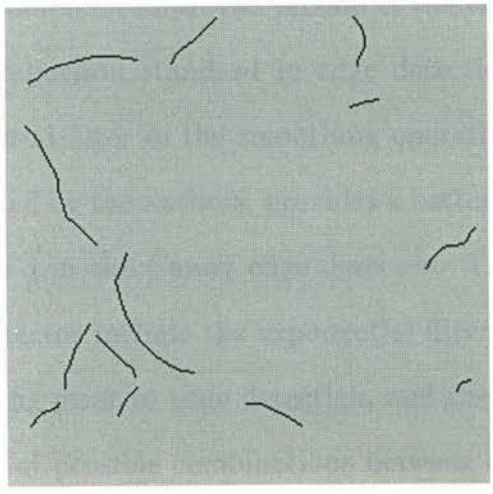
(a)

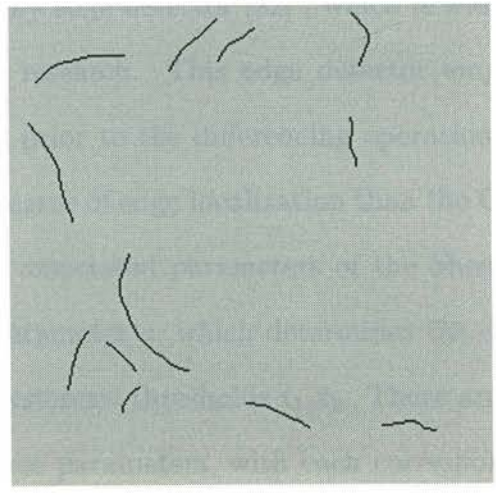




(b)

(c)





(d)

(e)

Figure 4.7: (a) Flower image (b)-(e) Edge examples supplied by human user.

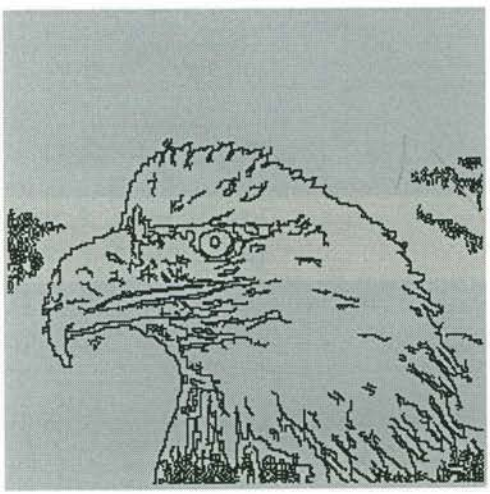
Figure 4.8: Edge tracings by human users in independent trials (solid curve: real edge, dashed curves: tracings by human users)

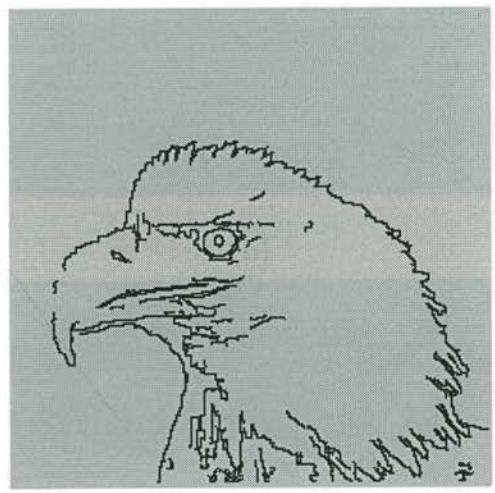
A comparison of Figure 4.9(a) with Figures 4.6(b) to (e) demonstrates the generalization capability of the NN-based edge detector: starting just from the acquired features in Figures 4.6(b) to (e), the network was able to locate other important edges in the eagle's image as perceived by humans. Although Figure 4.9(a) seems to be a satisfactory caricature of the original image, the performance of the edge characterizer has to be validated further by comparing the result with that of standard edge detectors. The performance of current edge detectors is usually controlled by a set of tunable parameters, and we would expect that the result of the NN edge detector should correspond to the performance of those edge detectors under near optimal settings of parameters.

We have compared the performance of the NN edge detector with that of the Shen-Castan edge detector [98] in Figures 4.9 and 4.10 : the Shen-Castan edge detector can be considered an improved version of the Canny edge detector [22] , which is widely used as a comparison standard in edge detection research. This edge detector employs an exponential filter in the smoothing operation prior to the differencing operation, which, as claimed by the authors, provides a better degree of edge localization than the Gaussian filter used in the Canny edge detector. The associated parameters of the Shen-Castan edge detector include the exponential filter parameter a, which determines the degree of smoothing prior to edge detection, and the hysteresis thresholds t_1, t_2. There are a large number of possible combinations between these parameters, with each corresponding to a very different resulting edge profile.

Figure 4.9 compares the NN feature detector result with the Shen-Castan edge detector under various settings of the hysteresis thresholds, while fixing the filter parameter a at

(a)





(b)

(c)

(d)

(e)

Figure 4.9: Edge detection results for the eagle image (a) Detected edges using NN. (b)-(e) Detected edges using Shen-Castan edge detector with different hysteresis thresholds t_1, t_2 (filter parameter $a = 0.3$) (b) $t_1 = 10, t_2 = 15$. (c) $t_1 = 20, t_2 = 25$. (d) $t_1 = 30, t_2 = 35$. (e) $t_1 = 40, t_2 = 45$.

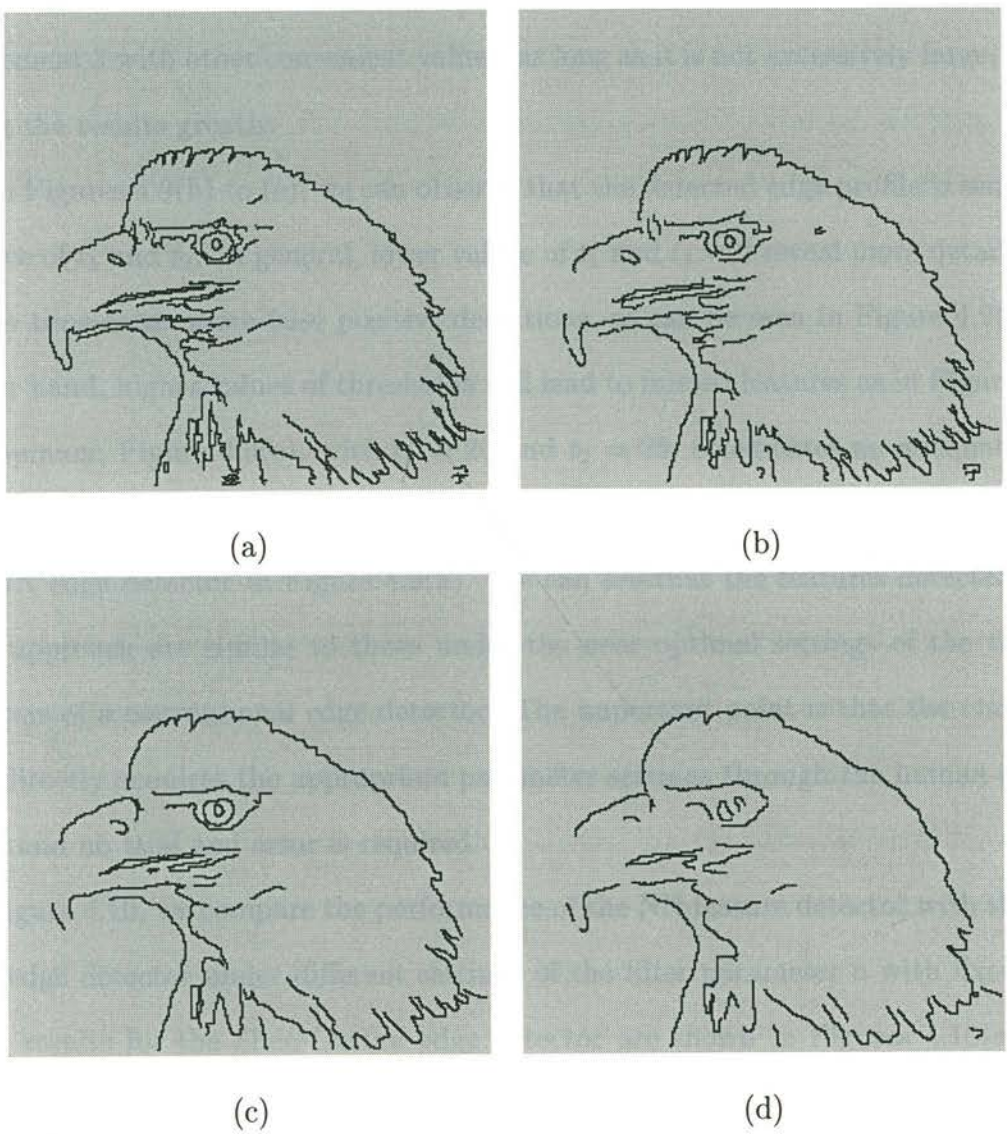


Figure 4.10: Edge detection results for the eagle image (a)-(d) Detected edges using Shen-Castan edge detector with different filter parameters a (hysteresis thresholds $t_1 = 20$, $t_2 = 25$) (a) $a = 0.4$. (b) $a = 0.5$. (c) $a = 0.6$. (d) $a = 0.7$.

0.3. The result of the NN edge detector is shown in Figure 4.9(a) and the results for the Shen-Castan edge detector are shown in Figures 4.9(b) to (e). The lower hysteresis threshold ranges from $t_1 = 10$ to $t_1 = 40$, and we have set the upper threshold $t_2 = t_1 + 5$. In general, the choice of this upper threshold t_2 is not as critical as the choice of the lower threshold t_1, as long as the condition $t_2 > t_1$ is satisfied. Therefore one can also replace the increment 5 with other convenient values, as long as it is not excessively large, without affecting the results greatly.

From Figures 4.9(b) to (e), we can observe that the detected edge profile is sensitive to the choice of t_1 and t_2. In general, lower values of t_1 and t_2 will reveal more details but at the same time cause more false positive detections, as can be seen in Figure 4.9(b) . On the other hand, higher values of thresholds will lead to missed features as in Figure 4.9(e). In our opinion, Figure 4.9(c), with $t_1 = 20$ and $t_2 = 25$, constitutes an adequate representation of the underlying features of the image. This can be compared with the result of the NN edge detector in Figure 4.9(a). We can see that the features detected by the current approach are similar to those under the near optimal settings of the threshold parameters of a conventional edge detector. The important point is that the current approach directly acquires the appropriate parameter settings through the human-specified features and no trial and error is required.

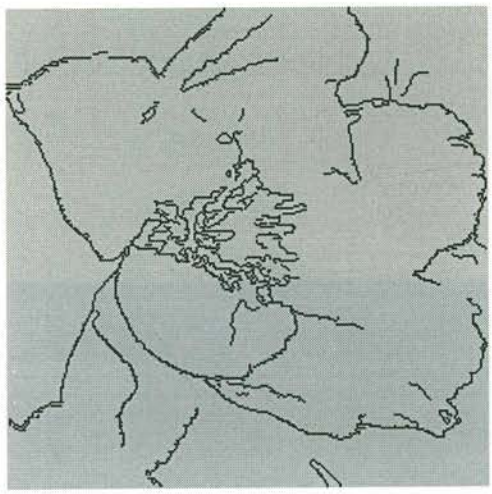
In Figure 4.10, we compare the performance of the NN feature detector with the Shen-Castan edge detector under different settings of the filter parameter a with fixed t_1 and t_2. The results for the Shen-Castan edge detector are shown in Figures 4.10(a) to (d) and we can compare them with the previous NN result in Figure 4.9(a). The settings for the filter width a range from $a = 0.4$ to $a = 0.7$. Although there is no corresponding concept of smoothing in the case of the NN detector, we can regard the setting of a as an alternative means to control the detection rate of false positives: in general, decreasing a will lead to more false positives, and increasing a will remove those false positives but lead to missed features. In addition, varying a will lead to the gradual shifting of the edges due to the smoothing action of the filter. We can observe all these effects as we increase a from Figures 4.10(a) to (d). In particular the shifting of the edges lend an

unnatural appearance to the corresponding edge profiles which is especially noticeable in Figures 4.10(c) and (d). The adjustment of this parameter is again avoided in the NN edge detector which directly assimilates the characteristics of human-defined features through the learning process.

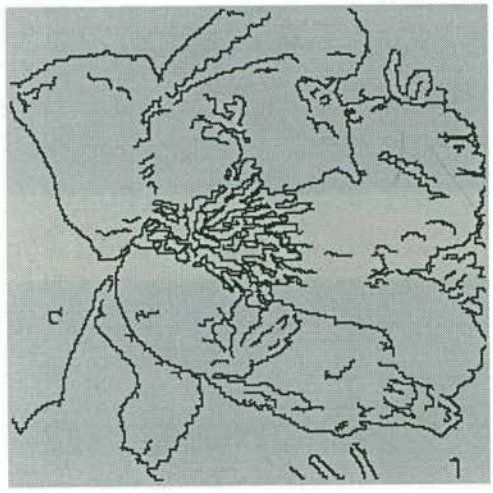
Comparison results for the flower image are shown in Figures 4.11 and 4.12, where Figure 4.11 shows the effect of varying the thresholds t_1, t_2, and Figure 4.12 shows the effect of varying the filter width a. The detected edges using the NN approach are shown in Figure 4.11(a). For the Shen-Castan edge detection results, we may consider Figure 4.11(c) (with thresholds $t_1 = 20$ and $t_2 = 25$) as a near optimal representation and Figure 4.11(d) as an adequate representation. Comparing these results with Figure 4.11(a), we notice that the quality of the NN detection results lies in between these two and thus can certainly be considered a close approximation to the optimal result. However, comparing Figure 4.11(a) with Figure 4.11(c), we see that the detected textures for NN are not as prominent as those in the Shen-Castan detection result. This can partly be attributed to the fact that no training examples have been taken from this area in Figures 4.7(b) to (e) and no explicit texture modelling has been implemented in the current model, and partly due to the particular value of threshold used in Figure 4.11(c) (we can compare this with the detected textures in Figure 4.11(d) and 4.11(e) under different thresholds). In general, we can expect the texture detection result to improve if we explicitly model the edges and textures as separate entities in the network [111].

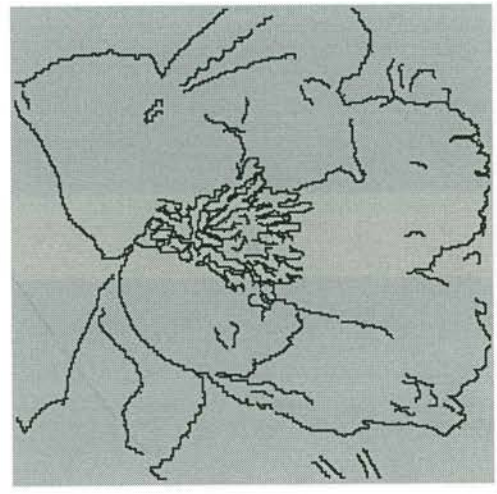
On the other hand, we may consider Figure 4.11(b) as over-cluttered with non-essential details, while missed features are clearly noticeable in Figure 4.11(e). In Figure 4.12, we notice the same effect of edge dislocation due to changes in the width parameter a. Again we emphasize that no process of trial and error in selecting parameters is required for the NN edge detector as compared with the Shen-Castan edge detector.

We further evaluated the generalization performance of the NN edge detector by applying the network trained on the eagle and flower features in Figures 4.6 and 4.7 to some other images. In Figures 4.13(a), image of a clock and other objects are shown, and Figure 4.13(c) shows an image with natural scenery. The corresponding detection results



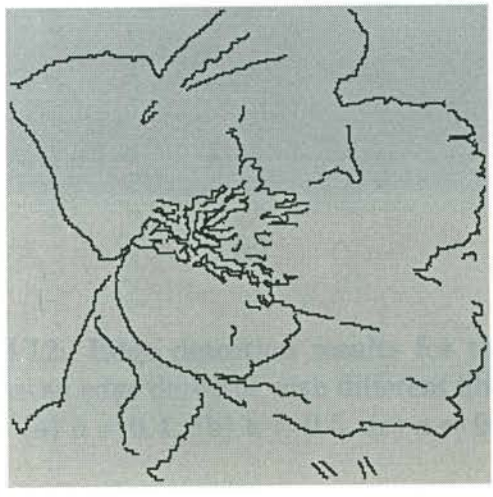
(a)

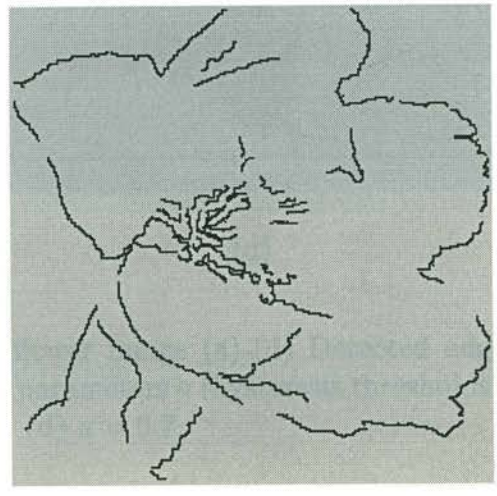




(b)

(c)





(d)

(e)

Figure 4.11: Edge detection results for the flower image (a) Detected edges using NN. (b)-(e) Detected edges using Shen-Castan edge detector with different hysteresis thresholds t_1, t_2 (filter parameter $a = 0.3$) (b) $t_1 = 10, t_2 = 15$. (c) $t_1 = 20, t_2 = 25$. (d) $t_1 = 30, t_2 = 35$. (e) $t_1 = 40, t_2 = 45$.

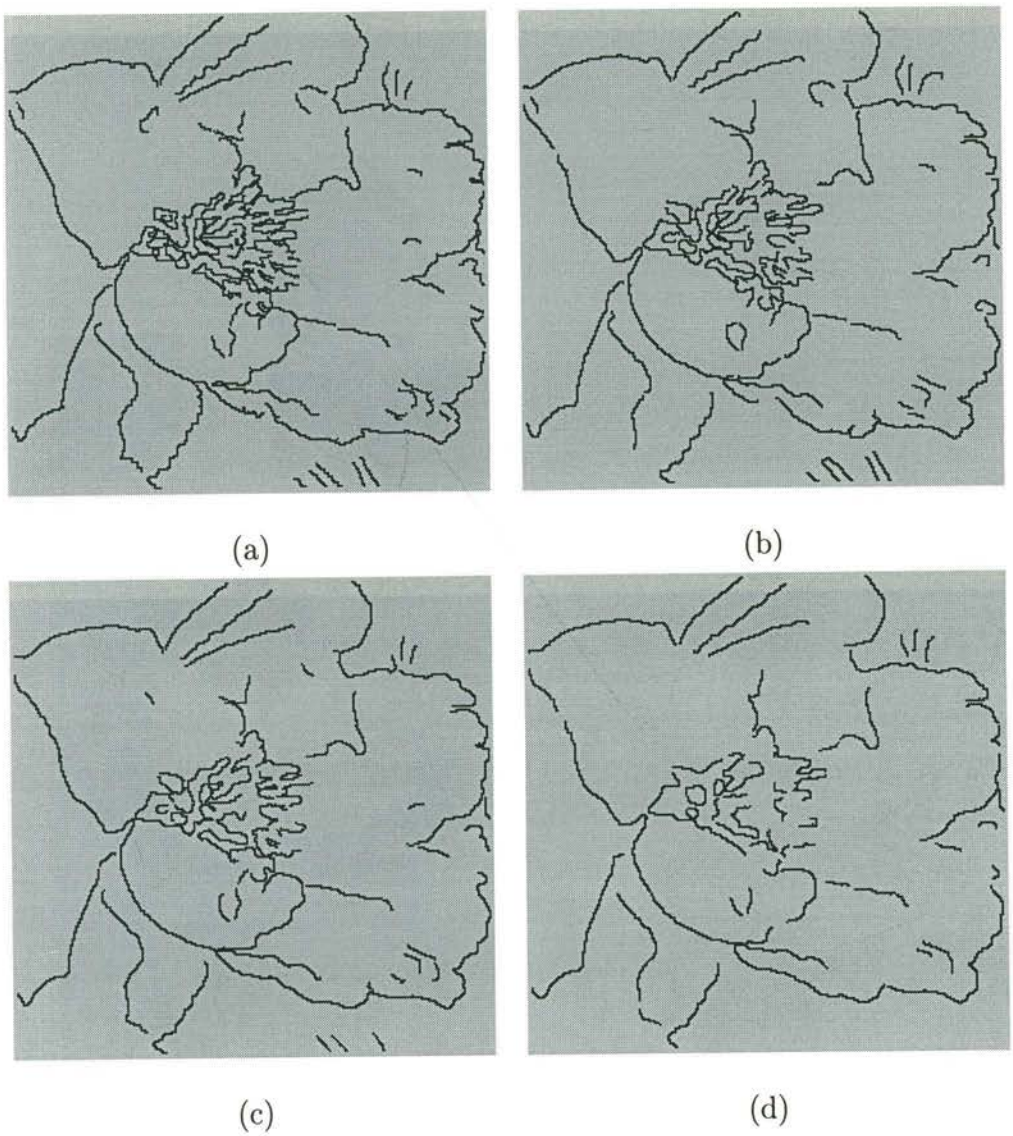


Figure 4.12: Edge detection results for the flower image (a)-(d) Detected edges using Shen-Castan edge detector with different filter parameters a (hysteresis thresholds $t_1 = 20$, $t_2 = 25$) (a) $a = 0.4$. (b) $a = 0.5$. (c) $a = 0.6$. (d) $a = 0.7$.

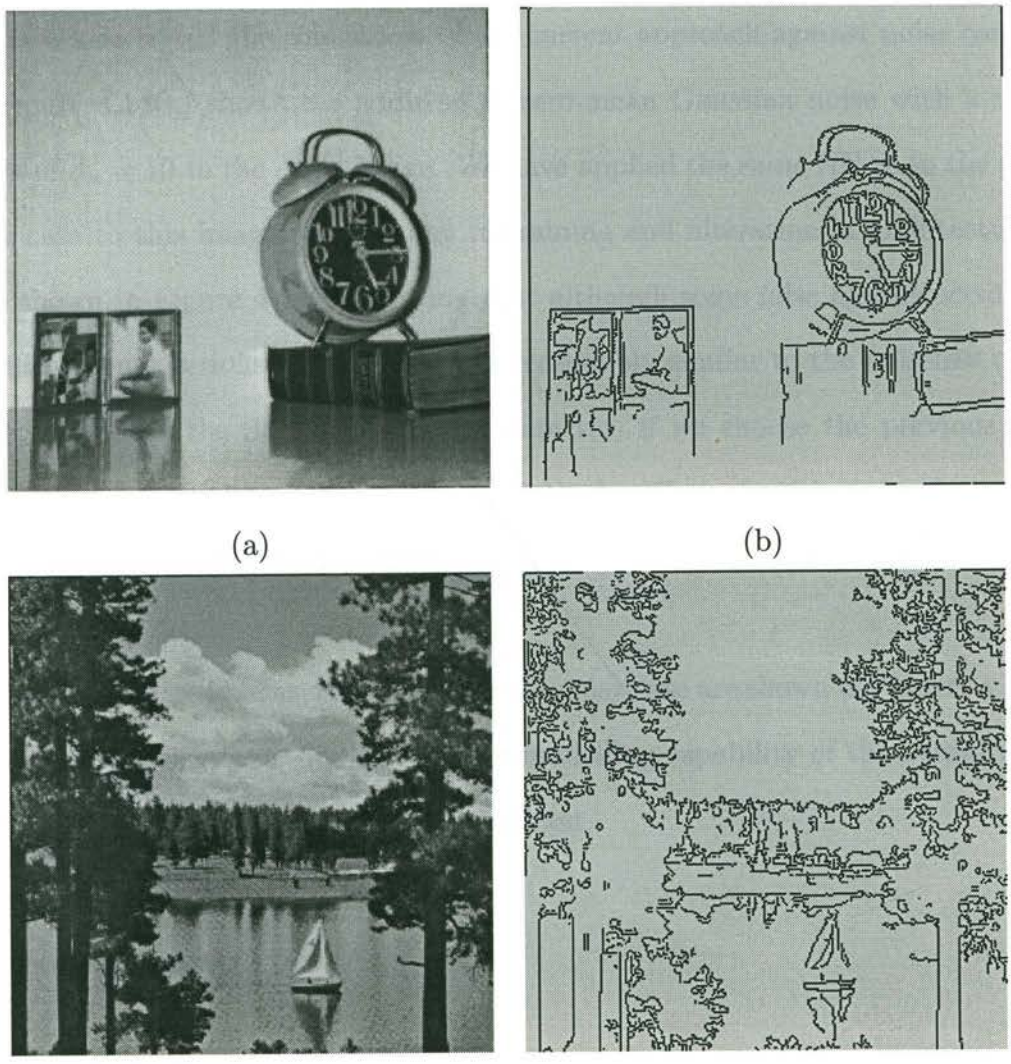


Figure 4.13: (a) Clock image (b) Detected edges using NN. (c) Natural scenery image (d) Detected edges using NN.

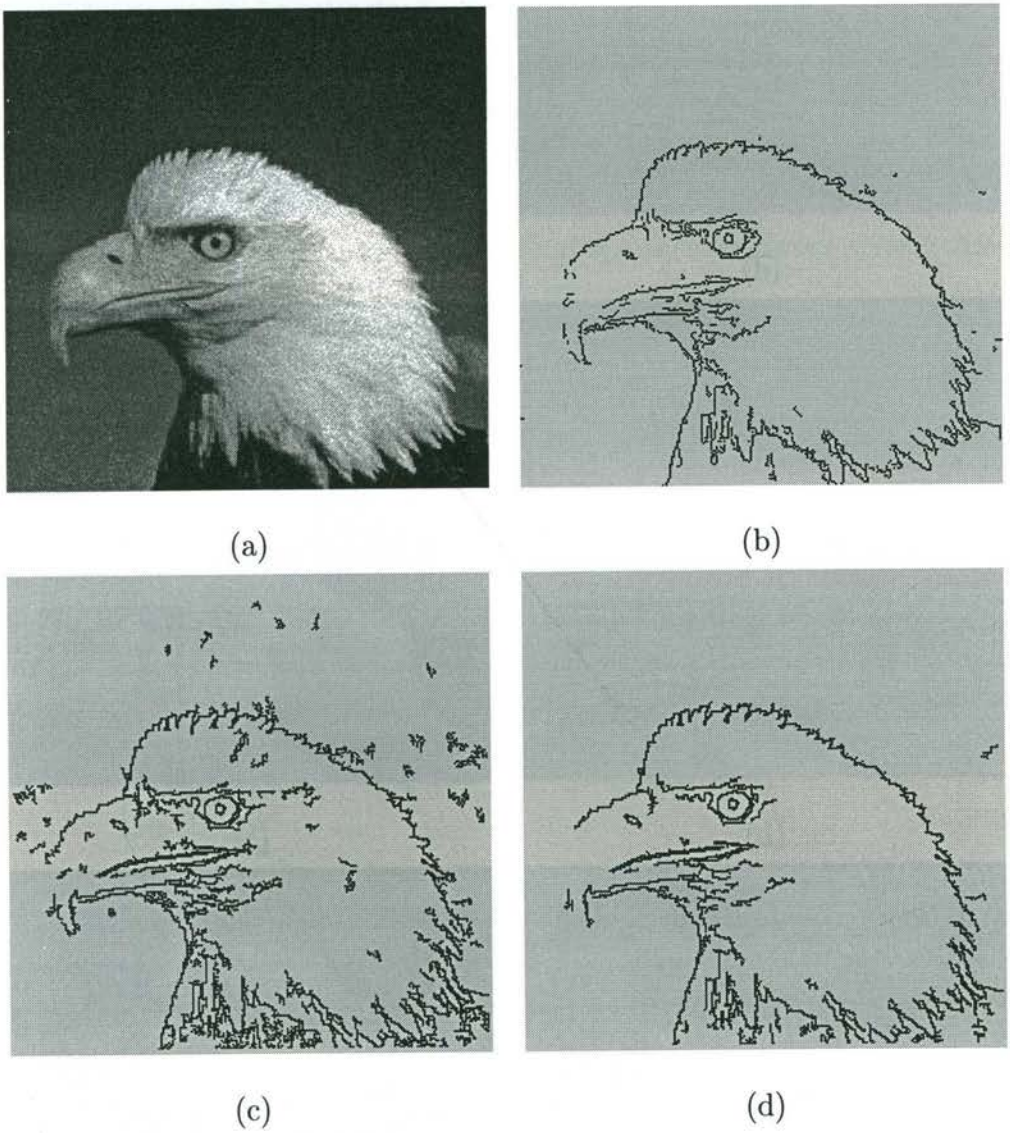
are shown in Figures 4.13(b) and (d) respectively . The results are very satisfactory, considering that the network is trained only on the eagle and flower images. The results also conform with some usual human preferences in interpreting pictures, such as the omission of clouds and the appearance of only the broad outline of trees, instead of the detailed textures within, in Figure 4.13(d).

We have also tested the robustness of the current approach against noise contaminations. Figure 4.14(a) shows the addition of zero-mean Gaussian noise with a standard deviation of $\sigma_n = 10$ to the eagle image. We have applied the same NN as in the previous noiseless case to this image without any re-training and alteration of architecture. The result is shown in Figure 4.14(b) showing that although some false alarms occurred, the overall effect is not serious and the result is reasonably similar to the noiseless case. On the other hand, for the Shen-Castan edge detector, if we choose the previous optimal threshold of $t_1 = 20$ and $t_2 = 25$ (Figure 4.14(c)), the effect of noise is clearly noticeable, and we will have to re-adjust the thresholds to $t_1 = 25$ and $t_2 = 30$ to eliminate its effect (Figure 4.14(d)).

For the other three images, the results for the noisy case are shown in Figures 4.15(b),(d) and (f). Again, notice that, due to the edge-modelling capability of the neural network, the effect of noise contamination is not significant.

4.6 Summary

We have developed a model-based feature detection neural network with hierarchical architecture (HMBNN) which directly assimilates the essential characteristics of human-specified features through a learning process. The specific architecture of the network divides the training features into sub-classes in such a way as to reflect the different preferences of human beings in regarding intensity discontinuities as features under different illumination conditions. Conventional edge detection algorithms using NN mainly focus on generalizing the front-end filtering operation while continuing to select the detection thresholds explicitly. The current HMBNN edge detector implicitly represents



(a) (b)

(c) (d)

Figure 4.14: (a) Eagle image with additive Gaussian noise ($\sigma_n = 10$) (b) Detected edges using NN. (c)-(d) Detected edges using Shen-Castan edge detector with different hysteresis thresholds t_1, t_2 (filter parameter $a = 0.3$). (c)$t_1 = 20, t_2 = 25$, (d) $t_1 = 25, t_2 = 30$

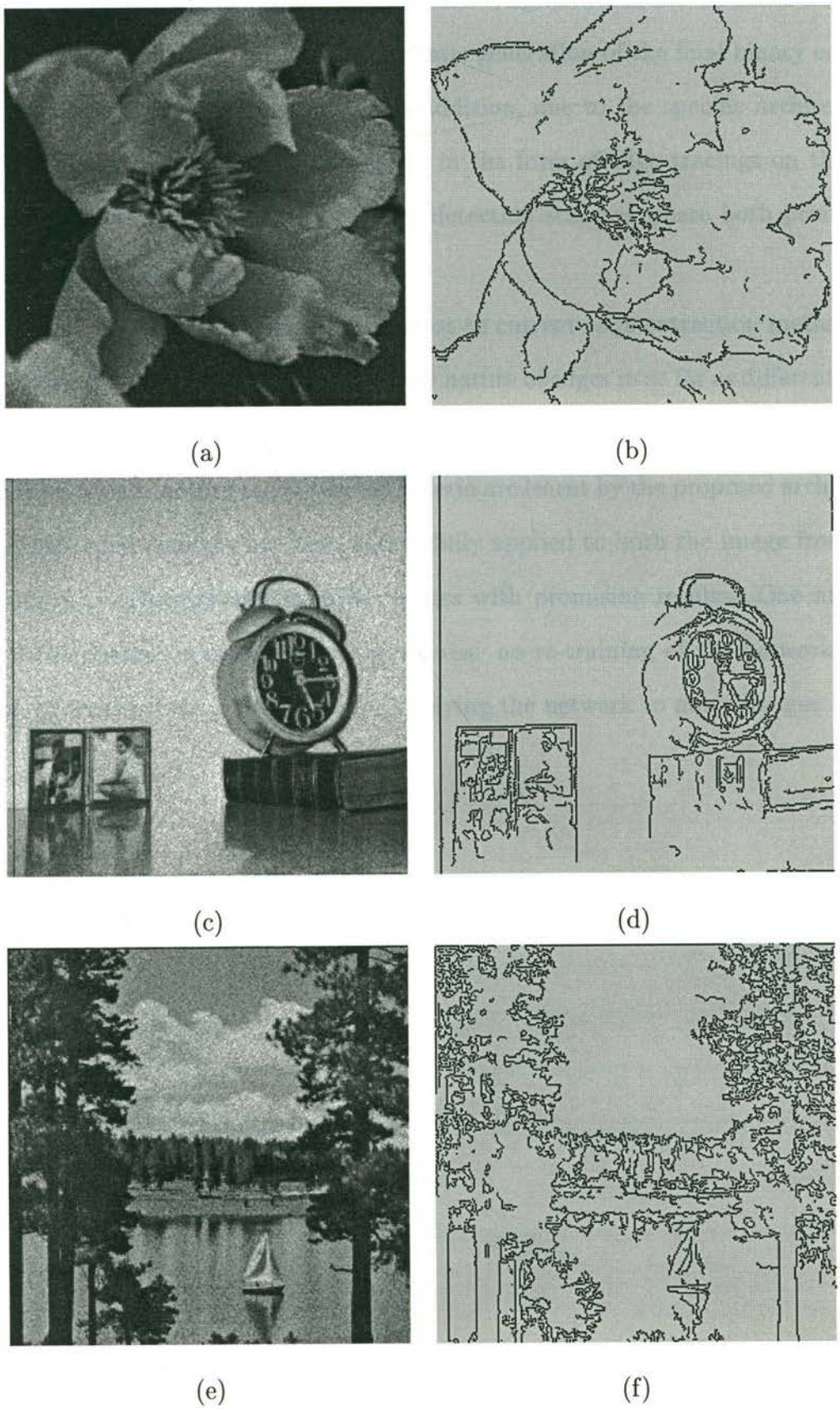


Figure 4.15: (a) Flower image with additive Gaussian noise ($\sigma_n = 10$) (b) Detected edges using NN. (c) Clock image with additive Gaussian noise ($\sigma_n = 10$) (d) Detected edges using NN. (e) Natural scenery image with additive Gaussian noise ($\sigma_n = 10$) (f) Detected edges using NN.

these decision parameters in the form of network weights which are updated during the training process, and which thus allow automatic generation of the final binary edge map without further parameter adjustments. In addition, due to the specific architecture of the network, only positive training examples in the form of edge tracings on the image are required, unlike previous NN-based edge detection schemes where both positive and negative examples are required.

It is also important to note that, in contrast to current edge-extraction methods, this HMBNN takes into account the non-stationary nature of edges in so far as different criteria apply to different image regions as a function of the local intensity distributions. The different rules which capture such different criteria are learnt by the proposed architecture. This HMBNN edge detector has been successfully applied to both the image from which its training data originate and to novel images with promising results. One attractive feature of the current approach is its robustness: no re-training of the network and no alteration of architecture are required for applying the network to noisy images.

Chapter 5

Distinguishing Between Edge and Texture: the Edge-Texture Characterization (ETC) Measure

5.1 Introduction

In this Chapter, we will introduce the Edge-Texture Characterization (ETC) measure, which is a scalar quantity summarizing the degree of resemblance of a particular pixel value configuration to either textures or edges. In other words, pixel value arrangements corresponding to textures and edges will in general exhibit different values for this measure. This is unlike the case where the local variance or the edge magnitude [50] is adopted for the image activity measure. Due to the possibility that both edges and textures may exhibit similar levels of image activities in terms of gray level variations around their neighborhoods, it is usually not possible to distinguish between these two feature types using the conventional image activity measures.

On the other hand, the current ETC measure is derived based on the correlational properties of individual pixel value configurations. In general, we may expect that configurations corresponding to edges and textures will possess significantly different correlational properties, with the individual pixels in a texture configuration being far less correlated

with each other than those in an edge configuration. This is described in a quantitative way using the current measure. More importantly, we have analytically established intervals of ETC measure values corresponding to those pixel configurations which visually more resemble textures than edges, and *vice versa.*

This measure is especially useful for distinguishing between edges and textures in image restoration such that we can specify different levels of regularization to each of them. Due to the different noise masking capabilities of these two feature types, it is usually not desirable to apply similar values of regularization parameters to both of them. This is seen in Chapter 3 where parameter values optimal to textured regions usually result in a noisy appearance for the edges due to its surrounding smooth regions, and those values suitable for edges usually cause blurring in textured regions. With the incorporation of the newly formulated ETC measure into our previous HMBNN approach in Chapter 6 , we are able to separately estimate two different parameter values which are optimal to edges and textures respectively, and apply the correct parameter to the current pixel in accordance with its associated ETC measure value.

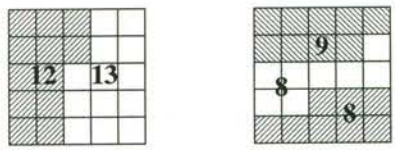
Even with the possibility of distinguishing between edges and textures, it is still possible to misclassify parts of the textured regions exhibiting weak variations as smooth regions, and apply corresponding large regularization values to those sub-regions. This will result in smooth blotches appearing within certain textured regions and lending an unnatural appearance to the overall image, as can be seen in some of the restored images under more severe degradations in Chapter 3. To remedy this problem, we have also derived a texture extraction algorithm based on the ETC measure, where the output of the algorithm are connected regions of textures established through the application of continuity constraint on those pixels with high ETC measure values, which most likely correspond to parts of textured regions. This *texture map* serves as an alternative source of information which complements our previous image activity classification map based on local variances, and is able to override incorrect classification of weak textured regions as smooth regions.

5.2 The Edge-Texture Characterization (ETC) Measure

The starting point of our formulation is as follows: we consider the gray level values of image pixels in a local region as i.i.d. random variables with variance σ^2. If we apply a local $K \times K$ averaging operation to each pixel, the variance σ'^2 of the smoothed random variables is given by

$$\sigma'^2 = \frac{1}{K^4}\mathbf{u}^T\mathbf{R}\mathbf{u} = \frac{\sigma^2}{K^2} \tag{5.1}$$

where $\mathbf{R} = diag[\sigma^2, \ldots, \sigma^2]$ is the $K^2 \times K^2$ covariance matrix of the K^2 random variables in the $K \times K$ averaging window, and $\mathbf{u}$ is an $K^2 \times 1$ vector with all entries equal to one. The diagonal structure of the covariance matrix is due to the independence of the random variables. The i.i.d. assumption above is in general not applicable to real-world images. In fact, we usually identify a meaningful image with the existence of controlled correlation among its pixels. As a result, we generalize the above i.i.d. case to incorporate correlations inside the $K \times K$ window.



P={12,13} (a) P={8,8,9} (b)

Figure 5.1: Illustrating the various forms of partition $\mathcal{P}$

We define the multiset $\mathcal{P} = \{P_1, \ldots, P_i, \ldots, P_m\}$, where $P_1 + \ldots + P_m = K^2$, as a *partition* of the K^2 variables in the window into m components. In addition, we assume that all variables within the i-th component is correlated with each other with correlation coefficient ρ_i, and variables among different components are mutually uncorrelated with each other. Some examples of $\mathcal{P}$ in a 5×5 window are given in Figure 5.1. For example, we can describe the region around an edge pixel by the partition $\mathcal{P} = \{P_1, P_2\}$, where $P_1 \approx P_2$ (Figure 5.1(a)). In this general case, the variance σ'^2 after $K \times K$ averaging is

given by

$$\sigma'^2 = \frac{1}{K^4}\mathbf{u}^T R u = \frac{\sigma^2}{\kappa^2} \tag{5.2}$$

In this case, $\mathbf{R}$ is a *block-diagonal* matrix with the following structure

$$\mathbf{R} = diag[\mathbf{R}_1, \ldots, \mathbf{R}_m] \tag{5.3}$$

Each sub-matrix $\mathbf{R}_i, i = 1, \ldots, m$ is of dimension $P_i \times P_i$ with the following structure

$$\mathbf{R}_i = \sigma^2 \begin{bmatrix} 1 & \rho_i & \ldots & \rho_i \\ \rho_i & \ddots & \ddots & \vdots \\ \vdots & \ddots & & \rho_i \\ \rho_i & \ldots & \rho_i & 1 \end{bmatrix} \tag{5.4}$$

If we carry out the matrix multiplication in equation (5.2), the square of the quantity κ in the equation is evaluated to be

$$\kappa^2 = \frac{K^4}{K^2 + \sum_{P_i \in \mathcal{P}} \rho_i (P_i^2 - P_i)} \tag{5.5}$$

Assuming that $0 \leq \rho_i \leq 1$ for all i, which implies positive correlation among pixels within a single component, the value of κ is maximized when $\rho_i = 0, \forall i$, giving $\kappa = K$ in equation (5.5), which corresponds to the previous case of i.i.d. variables.

On the other hand, if we assume $\rho_i = 1$ for all i within a single element partition $\mathcal{P} = \{K^2\}$ and substituting into equation (5.5), we have

$$\kappa^2 = \frac{K^4}{K^2 + (K^4 - K^2)} = 1 \tag{5.6}$$

which implies $\kappa = 1$. This corresponds to the case where all the gray level values within the window are highly correlated, or in other words, to smooth regions in the image. In general, the value of κ is between 1 and K. Thus it serves as an indicator of the degree of correlation within the $K \times K$ window. Larger values of κ indicate low level of correlation among the pixels in the window, which are usually the cases for textured and edge regions, while smaller values of κ usually correspond to smooth regions as indicated above. To provide an intuitive grasp of the values of κ corresponding to various features in the image, we carry out the calculation prescribed in equation (5.5) for a 5×5 averaging window.

For $K = 5$, the minimum and maximum values of κ are 1 and 5 respectively. For a positive within-component correlation coefficient ρ_i, the value of κ is constrained within the interval $[1, 5]$. Referring to Figure 5.1(a) which describes image edge regions with the partition $\mathcal{P} = \{12, 13\}$, and further assumes that $\rho_i = \rho = 0.9$ for all components, we have, after substituting the corresponding quantities into equation (5.5)

$$\kappa^2 = \frac{5^4}{5^2 + 0.9[(12^2 - 12) + (13^2 - 13)]} \approx 2.20 \tag{5.7}$$

or $\kappa \approx 1.48$. This value of κ, which we designate as κ_2, serves to characterize all edge-like features in the image if a 5×5 averaging window is used. On the other hand, if we consider more complex features with the number of components $m > 2$, which usually correspond to textures in the images, we should expect the value of κ to be within the interval $[1.48, 5]$. This is confirmed by evaluating κ for the partition $\mathcal{P} = \{8, 8, 9\}$ as illustrated in Figure 5.1(b), again assuming $\rho_i = 0.9$ for all i,

$$\kappa^2 = \frac{5^4}{5^2 + 0.9[2(8^2 - 8) + (9^2 - 9)]} \approx 3.28 \tag{5.8}$$

or $\kappa \approx 1.81$, which we designate as κ_3. As a result, the value of κ indicates to a certain extent the qualitative attributes of a local image region, i.e., whether it is more likely to be a textured region or an edge region, rather than just distinguishing the smooth background from the combined texture/edge regions as in the case of using the local standard deviation σ_{i_1,i_2} alone. We can therefore refer to this quantity κ as the Edge-Texture Characterization (ETC) measure.

κ_1	1.05	κ_6	2.54
κ_2	1.48	κ_7	2.72
κ_3	1.81	κ_8	2.91
κ_4	2.08	κ_9	3.07
κ_5	2.33	κ_{25}	5.00

Table 5.1: ETC measure values for various different partitions

Table 5.1 lists the values of the measure κ_1 to κ_9 and κ_{25}. We can notice that, in general, the value of κ increases with the number of correlated components within the pixel window, which confirms our previous observation. The case of 25 components corresponds to the case of i.i.d. random variables and results in the value $\kappa_{25} = 5$. It is also noted that the value of κ_1 is slightly larger than the case of fully correlated random variables in our previous calculation due to our present assumption of $\rho = 0.9$. For image processing applications in later chapters, we adopt the assumption that the intra-component correlation for smooth regions is greater than that for the edges and textures. As a result, while maintaining the value $\rho = 0.9$ for edges and textures, we will adopt $\rho = 1$ for smooth regions, which is equivalent to setting $\kappa_1 = 1$.

We can estimate the value of κ in a pixel neighborhood by the ratio $\hat{\sigma}/\hat{\sigma}'$ in accordance with equation (5.2) , where $\hat{\sigma}$ and $\hat{\sigma}'$ are the sample estimates of σ and σ' respectively.

$$\hat{\sigma}^2 = \frac{1}{|\mathcal{N}|} \sum_{(i,j)\in\mathcal{N}} (x_{i,j} - \overline{x})^2 \tag{5.9}$$

$$\hat{\sigma}'^2 = \frac{1}{|\mathcal{N}|} \sum_{(i,j)\in\mathcal{N}} (x'_{i,j} - \overline{x'})^2 \tag{5.10}$$

In the equations, $\mathcal{N}$ denotes a neighborhood set around the current pixel, $x_{i,j}$ denotes the gray level value of pixel (i, j) in the set, and $x'_{i,j}$ is the corresponding smoothed gray level value under $K \times K$ averaging. $\overline{x}$ and $\overline{x'}$ are respectively the mean of the gray level values of the original and smoothed variables in the neighborhood set. In general, this empirically estimated κ is not restricted to the interval $[1, K]$ due to the use of the sample variances, but most of its values are restricted to the interval $[0, K]$.

5.3 Experimental Results

To illustrate the capability of the ETC measure to distinguish between smooth regions, edges and textures, and especially between the latter two which cannot be easily separated using local variances, we present the *ETC-map*, which is a representation of the ETC values as a gray level image, of the images flower, Lena and eagle in Figures 5.2, 5.3 and 5.4. In addition, we highlight those image pixels with their corresponding κ values

within the following three intervals I_1, I_2 and I_3, where $I_1 = [0, \frac{\kappa_1+\kappa_2}{2}]$, $I_2 = (\frac{\kappa_1+\kappa_2}{2}, \kappa_3]$ and $I_3 = (\kappa_3, 5]$ in three separate binary maps. From the discussion in the previous section, pixels exhibiting κ values within the intervals I_1, I_2 and I_3 approximately correspond to smooth pixels, edge pixels and texture pixels respectively. This can be confirmed by visually inspecting the binary maps in Figures 5.2, 5.3 and 5.4. (The adoption of κ_3 rather than $\frac{\kappa_2+\kappa_3}{2}$ as the decision threshold between edge and texture is due to our observation that ,in terms of visual quality, the former constitutes a more appropriate transition point between the two feature types.)

Figure 5.2 shows the ETC-map and the feature segmentation maps for the flower image. From the ETC-map in Figure 5.2(a), it can be seen that the textured regions in the image appear brightest in the map, which correspond to the highest values of κ and is thus consistent with our previous assertion that texture pixels should exhibit higher values of κ. It is also seen that the edges correspond to intermediate values of κ, and the smooth regions appear darkest in the map, which again agree with our previous assertion.

To verify the characteristic κ values for the various feature types, we construct binary maps in Figures 5.2(b)-(d), where the black pixels are those with their κ values corresponding to the intervals I_1, I_2 and I_3 in Figures 5.2(b),(c) and (d) respectively. From the discussion in the previous section, it is expected that pixels with κ values in I_1 should form the smooth regions of the images, which is confirmed by visually inspecting Figure 5.2(b). Our expectation that those pixels with κ values within the intervals I_2 and I_3 should correspond to the edges and textures pixels respectively is also confirmed by inspecting Figures 5.2(c) and (d).

Figure 5.3 shows the same classifications performed on the image Lena. It can again be seen that specific ranges of ETC measure values are indicative of particular feature types in the image, thus confirming the discriminatory capability of this new measure. The same conclusion can be drawn by observing the results for the image eagle in Figure 5.4. The main feature types of this image consist of the smooth region and edge components as indicated in Figures 5.4(b) and (c). The texture components shown in Figure 5.4(d) are not as dominant as the other two components.

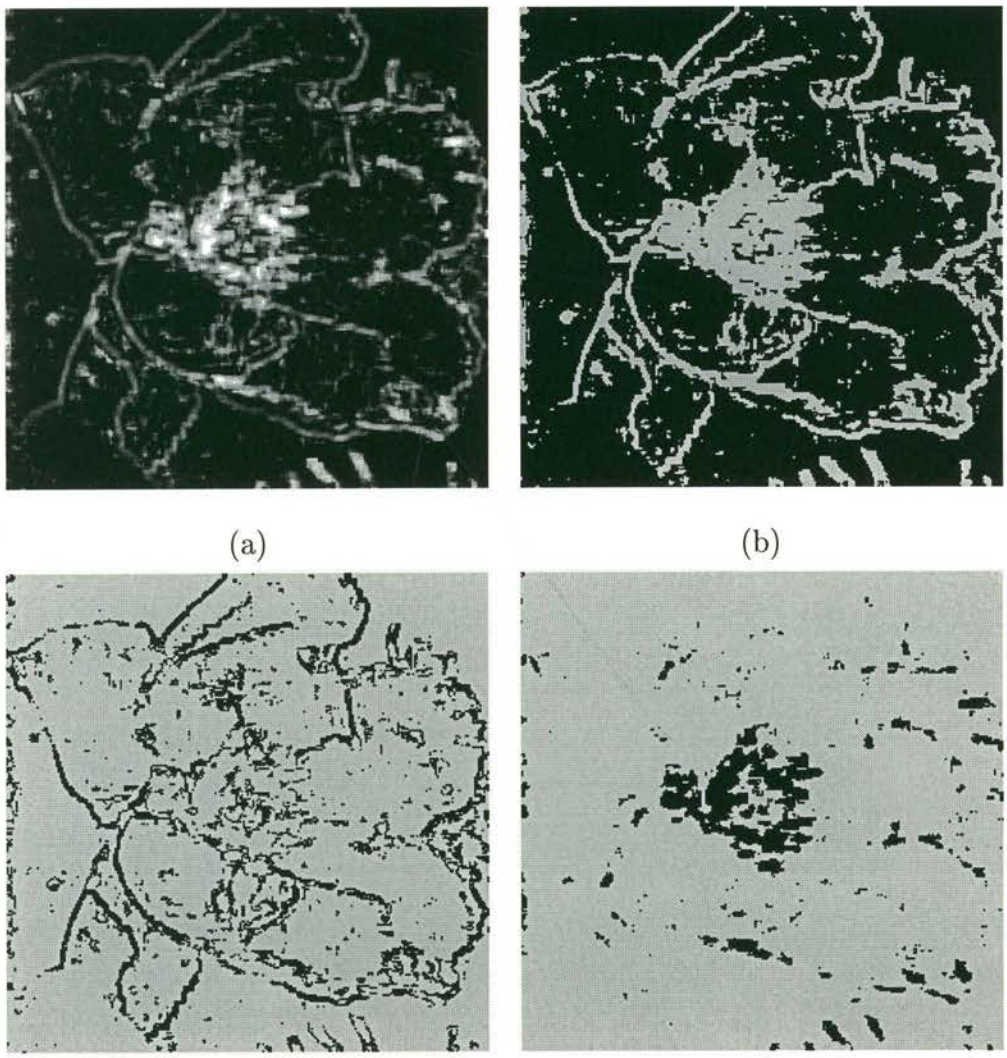


Figure 5.2: (a) ETC-map of flower (b) smooth region map (c) edge map (d) texture map

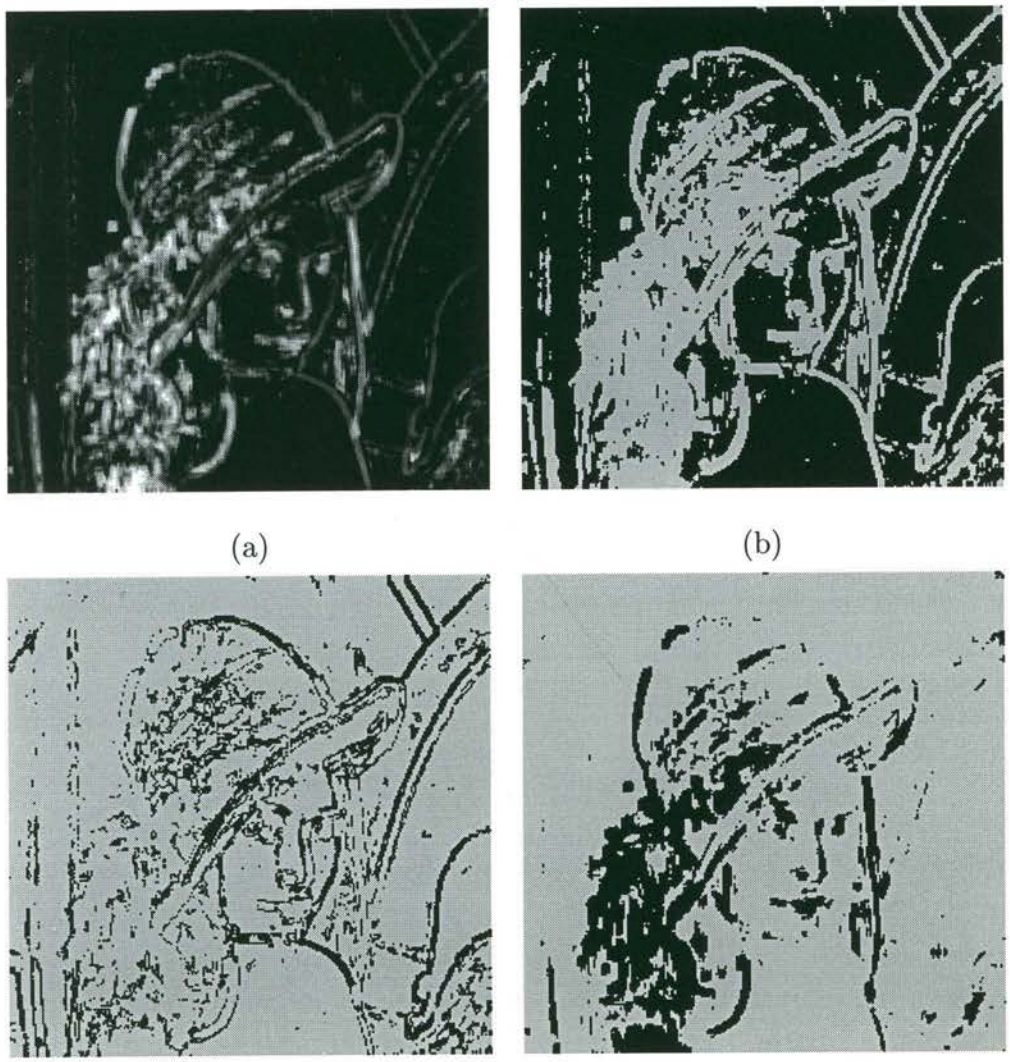


Figure 5.3: (a) ETC-map of Lena image (b) smooth region map (c) edge map (d) texture map

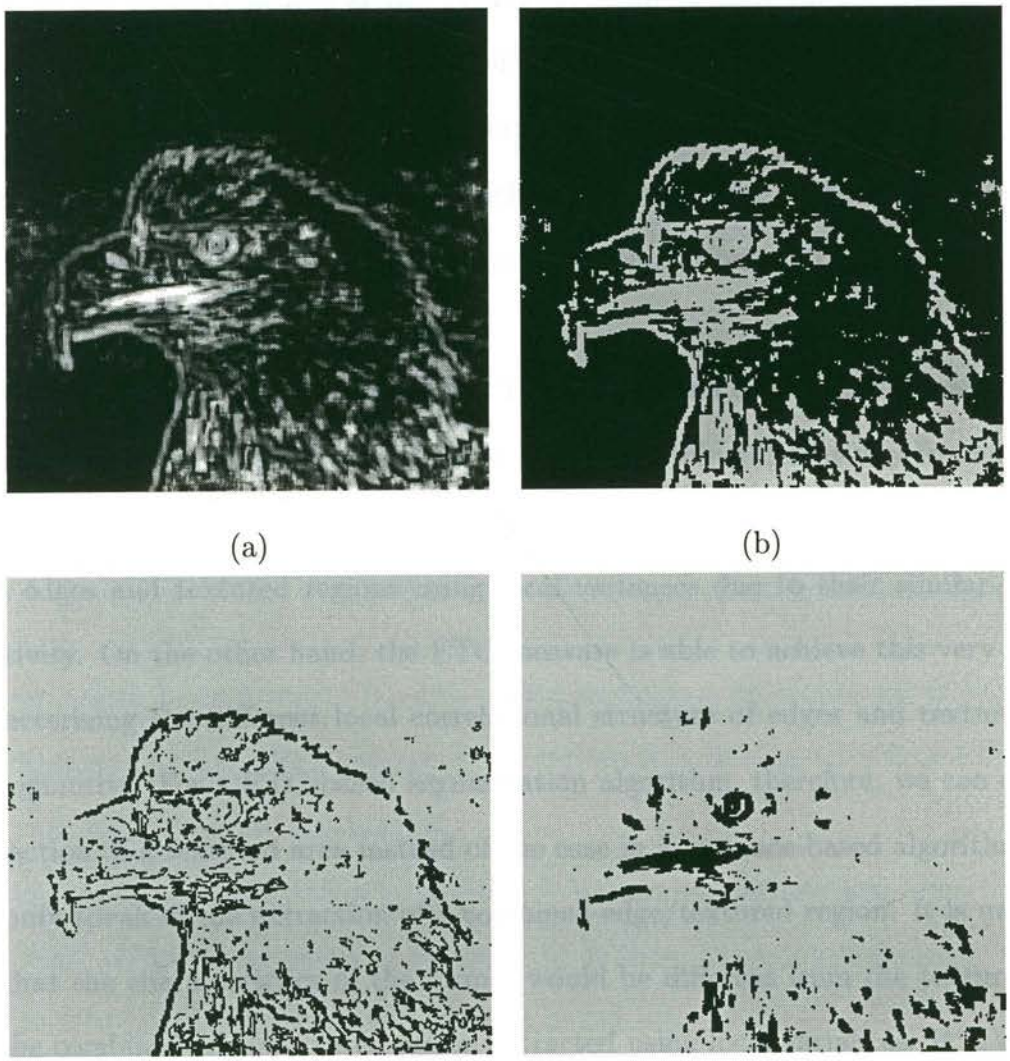


Figure 5.4: (a) ETC-map of eagle image (b) smooth region map (c) edge map (d) texture map

5.4 An algorithm for textured area extraction

In this section we describe an algorithm for textured area extraction based on the ETC measure. The motivation for this algorithm is due to our previous observation that, for severely degraded images, the HMBNN adaptive regularization algorithm produces blurred blotches within textured areas. This is the result of using the local pixel variances as the basis for the initial image segmentation, where the weak textural patterns within the area are classified as smooth regions. To overcome this problem, a complementary source of texture characterization information which is not dependent on the local variances is required. A possible candidate is a texture characterization map based on the newly formulated ETC measure.

It has been previously shown that the ETC measure can distinguish between smooth regions, edges and textures. This is achieved in a completely different way from using the local variances for similar purposes. For example, it is usually not possible to distinguish between edges and textured regions using local variances due to their similar levels of local activity. On the other hand, the ETC measure is able to achieve this very purpose by characterizing the different local correlational structure of edges and textures using a scalar quantity. For a ETC-based segmentation algorithm, therefore, we can speak of the extraction of a textured area instead of the case in a variance-based algorithm where we can only speak of the extraction of a combined edge/textured region. It is natural to expect that the characteristics of the former would be different from the texture subset within the combined edge/textured region extracted using local variances. It is therefore likely that the extracted textured area using ETC can provide a complementary source of texture characterization which can possibly eliminate the local blurring problem.

5.4.1 The Texture Extraction Algorithm

In view of the ability of the ETC measure to characterize textures in terms of a scalar quantity, we first apply a thresholding operation on the ETC values within an image. We then designate those pixels with κ greater than the threshold θ_κ as potential texture

pixels. From the discussion in the previous section, a reasonable choice for θ_κ would be approximately 1.8, but we have chosen $\theta_\kappa = 1.5$ to ensure that all the texture pixels are extracted at the expense of including some edge pixels, which we will remove in a later step.

Denoting the maximally connected components of the set of texture pixels as $R_a, a = 1, \ldots, A$, we apply two erosion operations [97], corresponding to the two structuring elements shown in Figure 5.5, to each of the regions. The structuring elements are designed such that horizontal or vertical sequences of pixels with run lengths less than 10 are removed, thus effectively excluding edge pixels from each region.

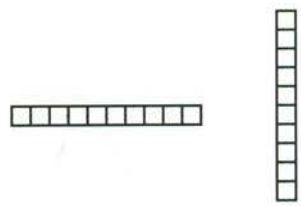


Figure 5.5: The two structuring elements used in the texture extraction algorithm

Corresponding to each region R_a, the erosion operations produce a set of sub-regions $U_{a,b_a} \subset R_a, b_a = 1, \ldots, B_a$. The number of sub-regions B_a will be greater than 1 if splitting of R_a occurs after the erosion operation.

In addition to removing the edge pixels, the erosion operations also remove some of the boundary pixels of the textured regions. To restore the region boundary, we follow the erosion with a modified dilation operation as follows.

For the region R_a, we denote the centroids of each associated sub-regions $U_{a,b_a}, b_a = 1, \ldots, B_a$ as $c_{a,b_a}, b_a = 1, \ldots, B_a$. Corresponding to each sub-region U_{a,b_a}, we define its *radius* r_{a,b_a} as follows:

$$r_{a,b_a} = \max |u - c_{a,b_a}| \qquad u \in U_{a,b_a} \tag{5.11}$$

For each region U_{a,b_a}, we define the dilated texture region U^D_{a,b_a} in terms of R_a and r_{a,b_a} as below

$$U^D_{a,b_a} = \{u \in R_a : |u - c_{a,b_a}| \leq r_{a,b_a}\} \tag{5.12}$$

By definition, the relation $U_{a,b_a} \subset U^D_{a,b_a}$ is always valid, but U^D_{a,b_a} also includes other pixels in R_a which are reasonably close to U_{a,b_a} but have been previously removed by the erosion operation, and in this way we restore part of the original boundaries of the textured regions. We designate the dilated regions $U^D_{a,b_a}, b_a = 1, \ldots, B_a$ as our final segmented textured regions. The above operations are illustrated in Figure 5.6.

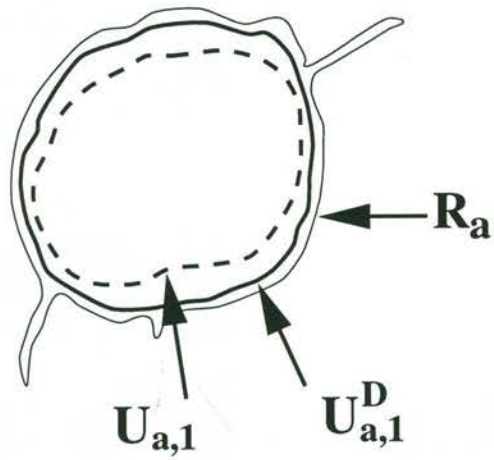
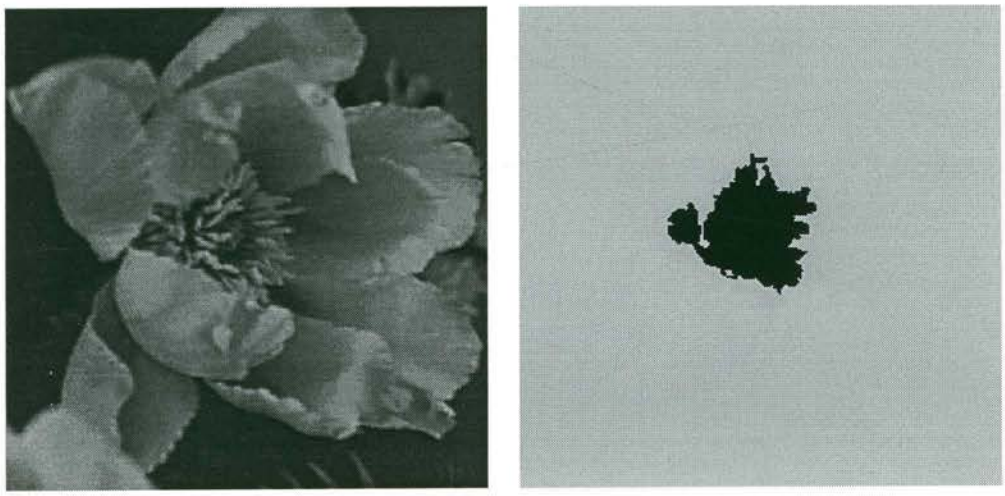


Figure 5.6: Illustrating the morphological operations for texture extraction

5.5 Experimental Results

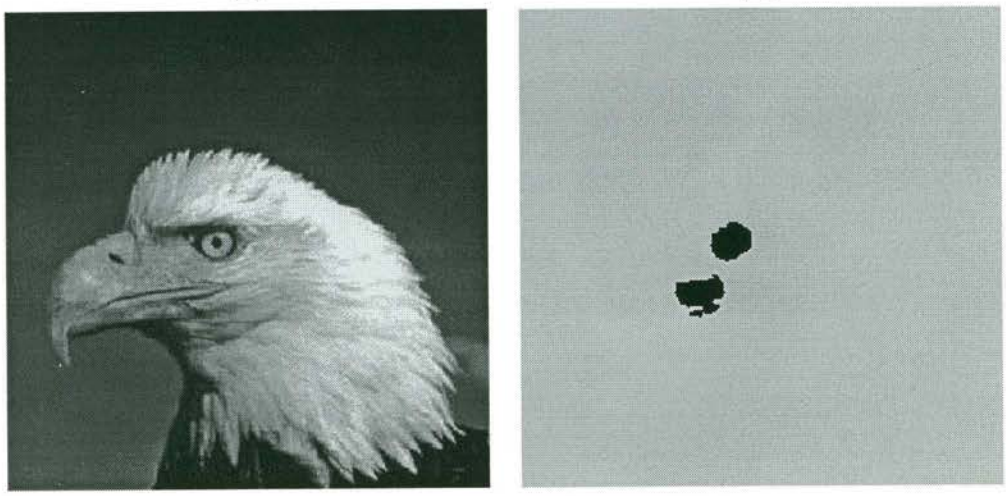
We have applied the texture extraction algorithm to the three images flower, Lena and eagle. For the images flower and Lena which contain substantial textures, we should expect the current algorithm to locate the corresponding regions approximately. For the image eagle which consists mainly of edges, it would still be insightful to observe what the algorithm would consider as textures under these circumstances.

The results are shown in Figure 5.7. We have included the original images flower, Lena and eagle in Figures 5.7(a), (c) and (e) respectively for easy comparison with the extraction results, which are shown in Figures 5.7(b), (d) and (f). The extracted regions for the image flower mainly concentrate around the flower stamen, which would usually be considered as textures by humans. For the Lena image, the regions are concentrated around the feathers of the hat, which again constitute textures in the image. We have thus



(a) (b)

(c) (d)



(e) (f)

Figure 5.7: Performance of textured region extraction algorithm (a) Flower image (b) Extracted textures in flower (c) Lena image(d) Extracted textures in Lena (e) Eagle image (f) Extracted textures in eagle.

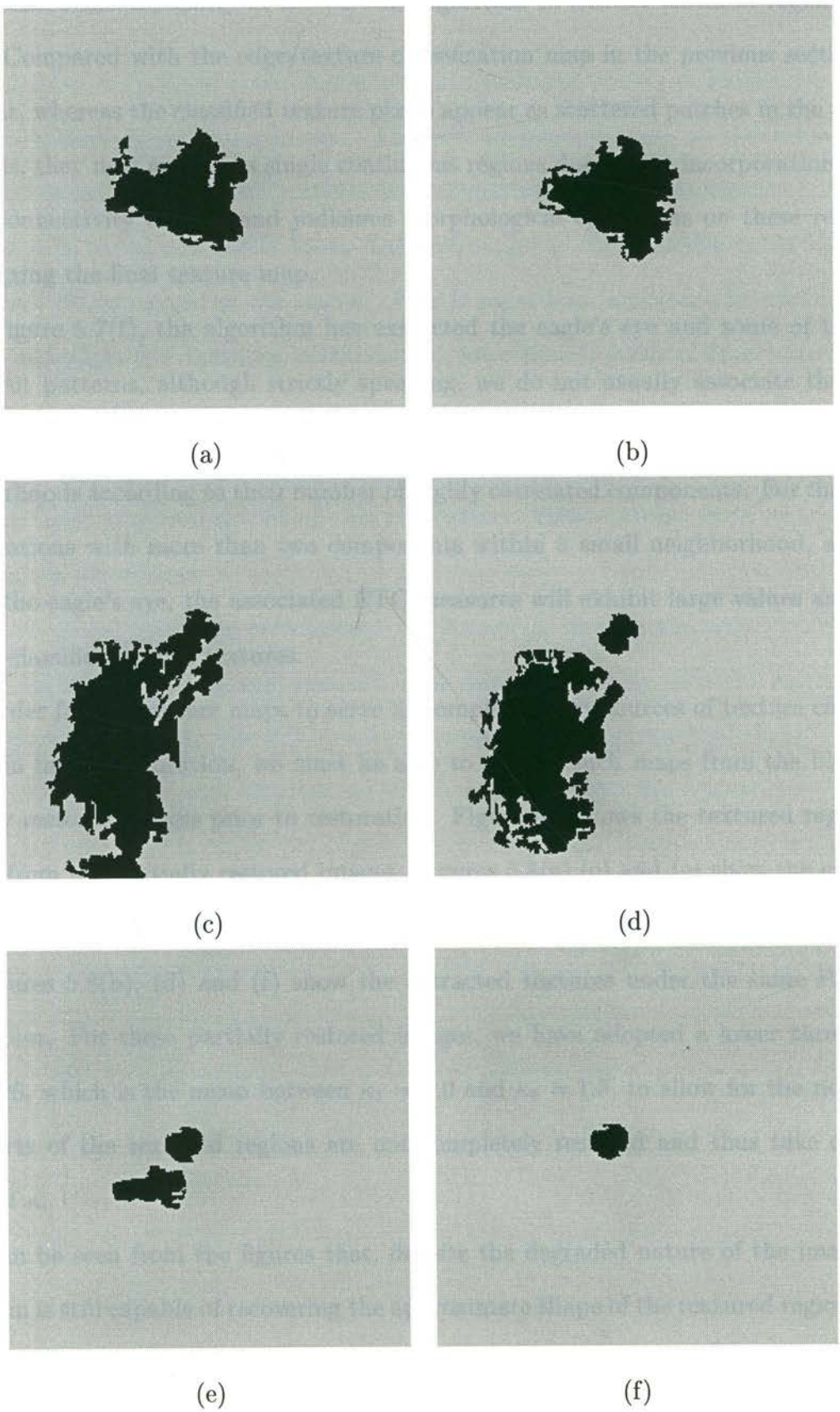


Figure 5.8: Texture extraction results for degraded images (a)-(b) Extracted textures in flower image (a) 30dB BSNR (b) 20dB BSNR. (c)-(d) Extracted textures in Lena image (c) 30dB BSNR (d) 20dB BSNR (e)-(f) Extracted textures in eagle image (e) 30dB BSNR (f) 20dB BSNR

demonstrated the capability of the current algorithm to extract textured regions in the image. Compared with the edge/texture classification map in the previous section, it is seen that, whereas the classified texture pixels appear as scattered patches in the previous examples, they now appear as single continuous regions due to our incorporation of both region connectivity criteria and judicious morphological operations on these regions in constructing the final texture map.

In Figure 5.7(f), the algorithm has extracted the eagle's eye and some of the more prominent patterns, although strictly speaking, we do not usually associate these with textural patterns. This is due to the nature of the ETC measure which classifies pixel neighborhoods according to their number of highly correlated components. For those pixel configurations with more than two components within a small neighborhood, as in the case of the eagle's eye, the associated ETC measures will exhibit large values and result in their classifications as textures.

In order for the texture maps to serve as complementary sources of texture characterization in image restoration, we must be able to extract such maps from the blurred or partially restored images prior to restoration. Figure 5.8 shows the textured regions extracted from the partially restored images. Figures 5.8(a),(c) and (e) show the extracted textures for the images flower, Lena and eagle under 5×5 uniform blur with 30dB noise, and Figures 5.8(b), (d) and (f) show the extracted textures under the same PSF with 20dB noise. For these partially restored images, we have adopted a lower threshold of $\theta_\kappa = 1.25$, which is the mean between $\kappa_1 = 1.0$ and $\kappa_2 = 1.5$, to allow for the possibility that parts of the textured regions are not completely restored and thus take on lower values of κ.

It can be seen from the figures that, despite the degraded nature of the images, the algorithm is still capable of recovering the approximate shape of the textured regions. This observation especially applies to the images flower and Lena where there are substantial areas of textures. Although we can notice some distortions to the region boundaries due to the degradations, the overall results can still be considered a valid characterization of the textures in the images.

5.6 Summary

In this Chapter, we propose a new characterization measure which can distinguish between the main feature types of an image. The new measure, known as the Edge-Texture Characterization measure (ETC), effectively summarizes the local correlational properties of an image in terms of a scalar value. Distinct ranges of this measure value correspond to different feature types of the image. This is especially important for distinguishing between the edges and textures, which usually have similar levels of local variances and cannot be separated using conventional methods. Based on this new measure, we have also derived a new texture extraction algorithm which specifically highlights the textured area as opposed to a combination of edge and texture pixels resulting from variance-based segmentation algorithms.

In view of the problems encountered in Chapter 3, namely the requirement to distinguish between the edges and textures of an image and apply different regularization strategies to each of them, the ETC measure represents an ideal candidate for the solution of this problem. In the next Chapter, we will characterize these two feature types in terms of two fuzzy sets defined on the set of ETC measure values. These are then incorporated into our previous HMBNN framework to form a fuzzified version of the algorithm.

Chapter 6

The ETC Fuzzy HMBNN for Adaptive Regularization

In this Chapter, we propose a fuzzified version of the model-based neural network for adaptive regularization in Chapter 3. A hierarchical architecture is again adopted for the current network, but we have included two neurons in each sub-network instead of a single neuron in Chapter 3. We referred to those two neurons as the edge neuron and the texture neuron respectively. They in turn estimate *two* regularization parameters, namely the edge parameter and the texture parameter, for each region with values optimal for regularizing the edge pixels and the textured pixels respectively within the region. Ideally, the parameter of the edge neuron should be updated using the information of the edge pixels only, and the texture neuron should be updated using the texture pixels only, which implies the necessity to distinguish between the edge and texture pixels. This is precisely the motivation for the formulation of the ETC measure in Chapter 5, which achieves this very purpose. As the concepts of edge and texture are inherently fuzzy as explained in Chapter 1, we characterize this fact by defining two fuzzy sets, the EDGE fuzzy set and the TEXTURE fuzzy set, over the ETC measure domain. This is possible since the value of the ETC measure reflects the degree of resemblance of a particular local gray level configuration to either textures or edges. More importantly, there exists definite intervals of measure values which we can claim that the corresponding gray level configuration

should be more like textures, and *vice versa*. As a result, the ETC measure domain serves as an ideal *universe of discourse* [61] for the definition of the EDGE and TEXTURE fuzzy sets. In view of the importance of the fuzzy set concept in this Chapter, we will first have a review on this topic.

6.1 Theory of Fuzzy Sets

The inclusion of a member a in a set A is usually represented symbolically as $a \in A$. Alternatively, we can express this inclusion in terms of a *membership function* $\mu_A(a)$ as follows

$$\mu_A(a) = \begin{cases} 1 & \text{if } a \in A \\ 0 & \text{if } a \notin A \end{cases} \tag{6.1}$$

The membership function takes values in the discrete set $\{0, 1\}$. Zadeh, in his 1965 paper [113] , generalized the definition of the membership function such that it can take values in the real-valued interval $[0, 1]$. The generalized set corresponding to the new membership function is known as a *fuzzy set* [59, 61, 113, 114, 116, 115] in contrast with the previous *crisp set*.

The implication of the new membership function is that, aside from the states of belonging wholly to a set or not belonging to a set at all, we can now allow an element to be a member of a fuzzy set "to a certain extent", with the exact degree of membership being expressed by the value of the corresponding membership function. Therefore, if the membership function value $\mu_A(a)$ of an element a is close to 1, we can say that a belongs "to a large degree" to the fuzzy set A, and we should expect that a possesses many of the properties that are characteristic of the set A. On the other hand, if the membership function value is small, we can expect that the element a only bears a vague resemblance to a typical member of the set A.

This concept is particularly important for expressing how human beings characterize everyday concepts. Take for example the concept "tall". We can immediately visualize the meaning of the word. But in order for machine interpretation systems to recognize

this concept, we must provide an exact definition. If restricted to the use of crisp sets, we may define this concept in terms of the set $T = \{h : h \geq 1.7m\}$ where h is the height variable. According to this definition, a value of the variable h such as $h = 1.699m$ will not be considered as belonging to the concept "tall", which is certainly unnatural from the human viewpoint. If we instead define "tall" to be a fuzzy set with the membership function shown in Figure 6.1, the value $h = 1.699m$ can still be interpreted as belonging "strongly" to the set, and thus conforms more closely to human interpretation.

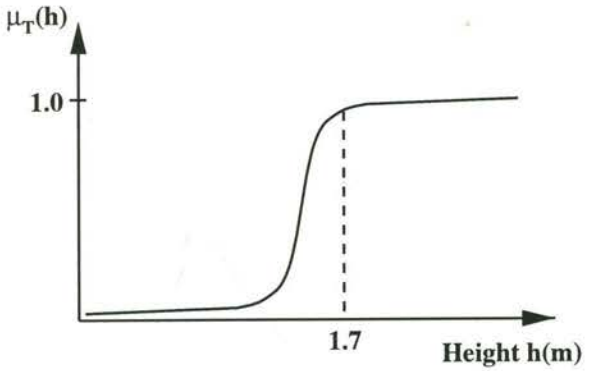


Figure 6.1: The fuzzy set representing the concept TALL

The ordinary operations on crisp set can be generalized to the case of fuzzy set in terms of the membership function values as follows:

$$\mu_{A\cap B}(x) = \min\{\mu_A(x), \mu_B(x)\} \tag{6.2}$$

$$\mu_{A\cup B}(x) = \max\{\mu_A(x), \mu_B(x)\} \tag{6.3}$$

$$\mu_{A^c}(x) = 1 - \mu_A(x) \tag{6.4}$$

where $A \cap B$, $A \cup B$ and A^c are respectively, the intersection of fuzzy sets A and B, the union of A and B, and the complement of A. These equations reduce to the ordinary intersection, union and complement operations on crisp sets when the ranges of the membership functions $\mu_A(x), \mu_B(x)$ are restricted to values in $\{0, 1\}$.

A *fuzzy inference relationship* is usually of the following form:

If $((x_1$ has property $A_1) \otimes \ldots\ldots \otimes (x_n$ has property $A_n))$, then $(y$ has property $B)$

where $x_1, \ldots, x_n$ and y are numerical variables, $A_1, \ldots, A_n$ and B are *linguistic* descriptions of the properties required of the corresponding numerical variables, and $\otimes$ denotes either the union or intersection operations. As described above, we can convert the linguistic descriptions $A_1, \ldots, A_n$ and B into fuzzy sets with membership functions $\mu_{A_1}(x_1), \ldots, \mu_{A_n}(x_n)$ and $\mu_B(y)$, where we have identified the names of the fuzzy sets with their corresponding linguistic descriptions. Formally, we can describe this inference operation as a mapping between fuzzy sets as follows:

$$B' = F(A_1, \ldots, A_n) \tag{6.5}$$

where $B' \subset B$ is a *fuzzy subset* of B. Fuzzy subsethood is formally described by the following condition

$$A \subset B \text{ if } \mu_A(x) \leq \mu_B(x), \forall x \tag{6.6}$$

which reduces to the ordinary subsethood relationship between crisp sets if $\mu_A(x), \mu_B(x)$ are allowed to take values only in $\{0, 1\}$.

The particular fuzzy subset B' chosen within the fuzzy set B (or equivalently, the particular form of the mapping F) depends on the degree to which the current value of each variable $x_i, i = 1, \ldots, n$ belongs to their respective fuzzy sets A_i. To summarize, the inference procedure accepts fuzzy sets as inputs and emits a single fuzzy set as output.

In practical systems, a *numerical* output which captures the essential characteristics of the output fuzzy set B' is usually required. This is usually done by specifying a *defuzzification* operation D on the fuzzy set B' to produce the numerical output y'

$$y' = D(B') \tag{6.7}$$

A common defuzzification operation is the following *centroid defuzzification* [61] operation

$$y' = \frac{\int_{-\infty}^{\infty} y \mu_{B'}(y)\, dy}{\int_{-\infty}^{\infty} \mu_{B'}(y)\, dy} \tag{6.8}$$

where we assign the centroid of the fuzzy membership function $\mu_{B'}(y)$ to the variable y'.

6.2 Edge-Texture Fuzzy Model Based on ETC measure

In Chapter 5, we have defined the Edge-Texture Characterization (ETC) measure κ which quantifies the degree of resemblance of a particular gray level configuration to either textures or edges. In addition, we have established that, for values of κ within the interval $I_2 = (\frac{\kappa_1+\kappa_2}{2}, \kappa_3]$, we can consider the underlying gray level configuration to be reasonably close to that of edges, and for $\kappa > \kappa_3$, we can conclude that the corresponding configuration has a closer resemblance to that of textures. However, if we consider the value $\kappa = \kappa_3 + \epsilon$, where ϵ is a small positive constant, we will classify the corresponding configuration as a texture configuration, but we can expect that it will still share many of the properties of an edge configuration due to the closeness of κ to an admissible edge value. In fact, it is difficult to define the concepts of "edge" and "textures" in terms of crisp sets in the ETC domain. In view of this, fuzzy set theory becomes a natural candidate for characterizing these concepts in terms of the ETC measure.

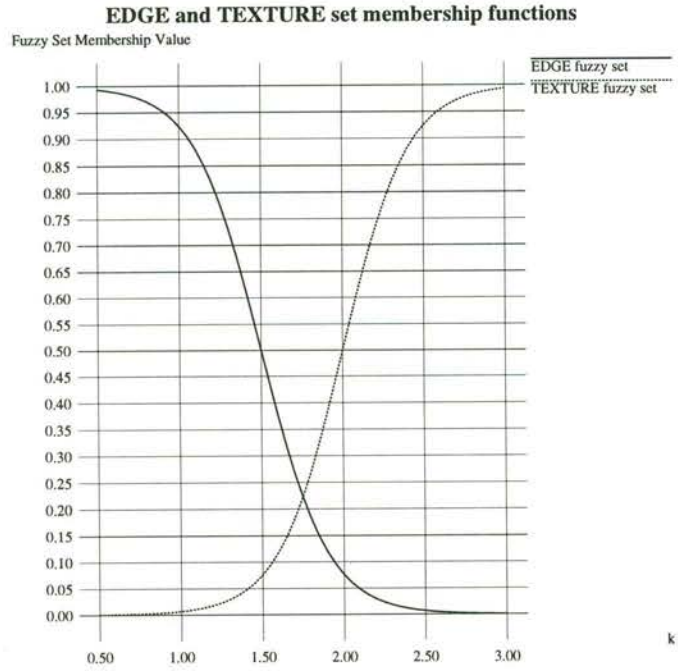


Figure 6.2: The EDGE and TEXTURE fuzzy membership functions

We therefore define two fuzzy sets, namely the EDGE fuzzy set and the TEXTURE fuzzy set, on the ETC measure domain in terms of their membership functions $\mu_E(\kappa)$ and $\mu_T(\kappa)$, as follows:

$$\mu_E(\kappa) \equiv \frac{1}{1+e^{\beta_E(\kappa-\kappa_E)}} \tag{6.9}$$

$$\mu_T(\kappa) \equiv \frac{1}{1+e^{-\beta_T(\kappa-\kappa_T)}} \tag{6.10}$$

The two set membership functions are plotted in Figure 6.2. From the figure, it is seen that $\mu_E(\kappa)$ is a decreasing sigmoid function with the transition point at $\kappa = \kappa_E$, and $\mu_T(\kappa)$ is an increasing sigmoid function with the transition point at $\kappa = \kappa_T$. The parameters β_E and β_T control the steepness of transition of the two membership functions. In view of the discussion in Chapter 5 which characterizes edges with values of κ within the interval I_2, we may have expected the function $\mu_E(\kappa)$ to exhibit a peak around κ_2 and tapers off on both sides. Instead, we have chosen a decreasing sigmoid function with the transition point at $\kappa_E \approx \kappa_2$, which implicitly classifies the smooth regions with $\kappa < \kappa_2$ as belonging to the EDGE fuzzy set as well. This is due to our formulation of the current regularization algorithm in such a way that, whenever a certain pixel configuration has a larger membership value in the EDGE fuzzy set, larger values of regularization parameters will be applied to this configuration. As a result, it is reasonable to assign greater membership values to those configurations with $\kappa < \kappa_2$, which usually corresponds to weak edges or smooth regions, due to their less effective noise masking capabilities compared with strong edges. In other words, we are interpreting the membership function $\mu_E(\kappa)$ as the indicator of the required amount of regularization for a certain pixel configuration rather than its true correspondence to a characteristic edge configuration.

On the other hand, the shape of the TEXTURE set membership function truly reflects the fact that we consider those gray level configurations with $\kappa > \kappa_3$ to be essentially textures. However, instead of choosing $\kappa_T = \kappa_3$, which was shown to correspond to a gray level configuration containing three uncorrelated components and may reasonably resemble textures, we instead choose the more conservative value of $\kappa_T = 2 > \kappa_3$.

From the two membership functions, we define the following *normalized ETC fuzzy*

coefficients $\tilde{\mu}_E(\kappa)$ and $\tilde{\mu}_T(\kappa)$

$$\tilde{\mu}_E(\kappa) = \frac{\mu_E(\kappa)}{\mu(\kappa)} \tag{6.11}$$

$$\tilde{\mu}_T(\kappa) = \frac{\mu_T(\kappa)}{\mu(\kappa)} \tag{6.12}$$

where

$$\mu(\kappa) = \mu_E(\kappa) + \mu_T(\kappa) \tag{6.13}$$

6.3 Architecture of the Fuzzy HMBNN

Corresponding to the partition of the image into the combined edge/texture components $\mathcal{F}_{r_f}, r_f = 1, \ldots, R_f$ and the smooth regions $\mathcal{B}_{r_b}, r_b = 1, \ldots, R_b$, we assign individual sub-networks to each of these regions. For the smooth regions, we assign ,as in Chapter 3, one neuron to each smooth region to estimate a single parameter $\lambda_{r_b}, r_b = 1, \ldots, R_b$ for each region. However, for the combined edge/textured regions, instead of employing only one neuron as in the previous case, we have assigned *two* neurons, which we designate as the edge neuron and the texture neuron with associated weight vectors $\tilde{\mathbf{p}}_{edge}(\lambda_{r_f}^{edge})$ and $\tilde{\mathbf{p}}_{tex}(\lambda_{r_f}^{tex})$, to each edge/texture sub-network. The two weight vectors are ,respectively, functions of the *edge regularization parameter* $\lambda_{r_f}^{edge}$ and the *texture regularization parameter* $\lambda_{r_f}^{tex}$. As their names imply, we should design the training procedure in such a way that the parameter $\lambda_{r_f}^{edge}$ estimated through the edge neuron should be optimal for the regularization of edge-like entities, and $\lambda_{r_f}^{tex}$ estimated through the texture neuron should be optimal for textured regions.

Corresponding to these two weight vectors, we evaluate two estimates of the required pixel change, $\triangle\tilde{v}_{i_1,i_2}^{edge}$ and $\triangle\tilde{v}_{i_1,i_2}^{tex}$, for the same gray level configuration $\tilde{\mathbf{v}}_{i_1,i_2}$ as follows

$$\triangle\tilde{v}_{i_1,i_2}^{edge} = \tilde{\mathbf{p}}_{edge}(\lambda_{r_f}^{edge})^T\tilde{\mathbf{v}}_{i_1,i_2} \tag{6.14}$$

$$\triangle\tilde{v}_{i_1,i_2}^{tex} = \tilde{\mathbf{p}}_{tex}(\lambda_{r_f}^{tex})^T\tilde{\mathbf{v}}_{i_1,i_2} \tag{6.15}$$

where $(i_1, i_2) \in \mathcal{F}_{r_f}$. The quantities $\triangle\tilde{v}_{i_1,i_2}^{edge}$ and $\triangle\tilde{v}_{i_1,i_2}^{tex}$ are the required updates based on the assumptions that the underlying gray level configuration $\tilde{\mathbf{v}}_{i_1,i_2}$ corresponds to edge or textures respectively.

Referring to Chapter 5 and the fuzzy formulation in the previous section, we can evaluate the ETC measure κ_{i_1,i_2} at pixel position (i_1, i_2) as a function of the pixel values in a local neighborhood of (i_1, i_2). From this we can calculate the normalized edge fuzzy coefficient value $\widetilde{\mu}_E(\kappa_{i_1,i_2})$ and the normalized texture fuzzy coefficient value $\widetilde{\mu}_T(\kappa_{i_1,i_2})$. As a result, a natural candidate for the final required pixel update value $\Delta\widetilde{v}_{i_1,i_2}$ can be evaluated as a convex combination of the quantities $\Delta\widetilde{v}^{edge}_{i_1,i_2}$ and $\Delta\widetilde{v}^{tex}_{i_1,i_2}$ with respect to the normalized fuzzy coefficient values

$$\Delta\widetilde{v}_{i_1,i_2} = \widetilde{\mu}_E(\kappa_{i_1,i_2})\Delta\widetilde{v}^{edge}_{i_1,i_2} + \widetilde{\mu}_T(\kappa_{i_1,i_2})\Delta\widetilde{v}^{tex}_{i_1,i_2} \tag{6.16}$$

From the above equation, it is seen that for regions around edges where $\widetilde{\mu}_E(\kappa_{i_1,i_2}) \approx 1$ and $\widetilde{\mu}_T(\kappa_{i_1,i_2}) \approx 0$, the final required gray level update $\Delta\widetilde{v}_{i_1,i_2}$ is approximately equal to $\Delta\widetilde{v}^{edge}_{i_1,i_2}$, which is optimal for the restoration of the edges. On the other hand, in textured regions where $\widetilde{\mu}_T(\kappa_{i_1,i_2}) \approx 1$ and $\widetilde{\mu}_E(\kappa_{i_1,i_2}) \approx 0$, the required update $\Delta\widetilde{v}_{i_1,i_2}$ assumes the value $\Delta\widetilde{v}^{tex}_{i_1,i_2}$ which is optimal for textures, provided that proper training procedures are adopted for both of the neurons. Alternatively, this equation can be interpreted as the operation of a *two-layer* sub-network, where $\Delta\widetilde{v}_{i_1,i_2}$ is the network output, $\widetilde{\mu}_E(\kappa_{i_1,i_2}), \widetilde{\mu}_T(\kappa_{i_1,i_2})$ are output weights, and $\Delta\widetilde{v}^{edge}_{i_1,i_2}$ and $\Delta\widetilde{v}^{tex}_{i_1,i_2}$ are the hidden neuron outputs. The architecture of this two-layer sub-network is shown in Figure 6.3.

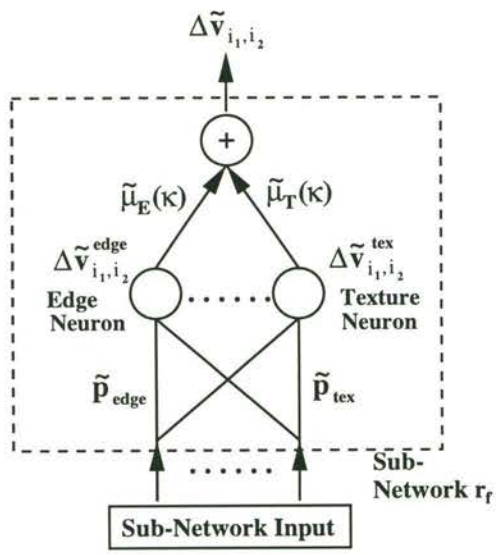


Figure 6.3: The architecture of the fuzzy HMBNN sub-network

6.3.1 Correspondence with the General HMBNN Architecture

Similar to our previous model-based neural network formulation in Chapter 3, the current fuzzy HMBNN model also adopts the hidden-node hierarchical architecture, which is depicted in Figure 3.1 in Chapter 3. The architecture of the sub-networks for the regularization of the smooth regions remains the same. However, for the edge/texture regularization sub-networks, we have adopted a new architecture in the form of a two-layer network, with two neurons in the hidden layer as shown in Figure 6.3. Corresponding to the general form for evaluation of the sub-network output in Chapter 2, we can describe the operation of the edge/texture regularization sub-network for region $\mathcal{F}_{r_f}$ as follows:

$$\phi(\tilde{\mathbf{v}}_{i_1,i_2}, \tilde{\mathbf{p}}_{r_f}) = \sum_{s=1}^{2} c_s \psi_{r_f}(\tilde{\mathbf{v}}_{i_1,i_2}, \tilde{\mathbf{p}}(\lambda_{r_f}^s)) \tag{6.17}$$

where

$$\lambda_{r_f}^1 \equiv \lambda_{r_f}^{edge} \tag{6.18}$$

$$\lambda_{r_f}^2 \equiv \lambda_{r_f}^{tex} \tag{6.19}$$

$$\psi_{r_f}(\tilde{\mathbf{v}}_{i_1,i_2}, \tilde{\mathbf{p}}(\lambda_{r_f}^s)) \equiv \tilde{\mathbf{p}}(\lambda_{r_f}^s)^T \tilde{\mathbf{v}}_{i_1,i_2} \tag{6.20}$$

$$c_1 \equiv \tilde{\mu}_E(\kappa_{i_1,i_2}) \tag{6.21}$$

$$c_2 \equiv \tilde{\mu}_T(\kappa_{i_1,i_2}) \tag{6.22}$$

and $\tilde{\mathbf{p}}_{r_f}$ can be considered as a concatenation of $\tilde{\mathbf{p}}(\lambda_{r_f}^s), s = 1, 2$.

6.4 Estimation of the Desired Network Output

In Chapter 3, we define the desired network output $\triangle\tilde{v}_{i_1,i_2}^d$ and the associated predicted gray level value $\tilde{v}_{i_1,i_2}^d$, where

$$\tilde{v}_{i_1,i_2}^d = \tilde{v}_{i_1,i_2} + \triangle\tilde{v}_{i_1,i_2}^d \tag{6.23}$$

as follows: for smooth regions, we define the variable $\tilde{v}_{i_1,i_2}^d$ as the mean of the neighboring pixel values

$$\tilde{v}_{i_1,i_2}^d = \bar{\tilde{v}}_{i_1,i_2} \tag{6.24}$$

$$\bar{\bar{v}}_{i_1,i_2} = \frac{1}{N_p} \sum_{k=-L_p}^{L_p} \sum_{l=-L_p}^{L_p} \tilde{v}_{i_1+k,i_2+l} \tag{6.25}$$

where $N_p = (2L_p + 1)^2$ is the size of the filter mask for averaging. For combined edge/textured regions, we consider the ranked pixel values in the prediction neighborhood set

$$\mathcal{N}_p = \{\tilde{v}^{(1)}_{i_1,i_2}, \ldots, \tilde{v}^{(n_p)}_{i_1,i_2}, \ldots, \tilde{v}^{(N_p)}_{i_1,i_2}\}$$

where $\tilde{v}^{(n_p)}_{i_1,i_2}$ is the n_p-th order statistic of the pixels in the ordered prediction neighborhood set and corresponds to $\tilde{v}_{i_1+k,i_2+l}$ for appropriate k and l. The ranked gray level values satisfy the following condition

$$\tilde{v}^{(1)}_{i_1,i_2} \leq \ldots \leq \tilde{v}^{(n_p)}_{i_1,i_2} \leq \ldots \leq \tilde{v}^{(N_p)}_{i_1,i_2}$$

In Chapter 3, adopting a prediction neighborhood size of $N_p = 9$, we define the predicted gray level value $\tilde{v}^d_{i_1,i_2}$ for the combined edge/textured regions as follows:

$$\tilde{v}^d_{i_1,i_2} = \begin{cases} \tilde{v}^{(3)}_{i_1,i_2} & \tilde{v}_{i_1,i_2} < \bar{\bar{v}}_{i_1,i_2} \\ \tilde{v}^{(7)}_{i_1,i_2} & \tilde{v}_{i_1,i_2} \geq \bar{\bar{v}}_{i_1,i_2} \end{cases} \tag{6.26}$$

It was found from the experiments in Chapter 3 that this *texture-oriented* estimation scheme is appropriate for restoring images at moderate noise levels, and for the textured areas in restored images at high noise levels. However, the resulting predicted gray level value will lead to a noisy appearance for the edges at high noise levels due to its smaller noise masking capability compared with textures. In view of this, we can apply the alternative *edge-oriented* estimation scheme at high noise levels

$$\tilde{v}^d_{i_1,i_2} = \begin{cases} \tilde{v}^{(4)}_{i_1,i_2} & \tilde{v}_{i_1,i_2} < \bar{\bar{v}}_{i_1,i_2} \\ \tilde{v}^{(6)}_{i_1,i_2} & \tilde{v}_{i_1,i_2} \geq \bar{\bar{v}}_{i_1,i_2} \end{cases} \tag{6.27}$$

This edge-oriented scheme results in a less noisy appearance for the edges, but the textured areas will appear blurred compared with the previous case. It would therefore be ideal if, at high noise levels, we can apply the texture-oriented estimation scheme to the textured areas only, and apply the edge-oriented estimation scheme to only the edges. This in turn requires the separation of the edges and textured areas, which is usually difficult in

terms of conventional measures such as image gradient magnitudes or local variance due to the similar levels of gray level activities for these two types of features. On the other hand, the previous fuzzy formulation in terms of the scalar ETC measure allows this very possibility.

In addition, equations (6.26) and (6.27) predict the desired gray level value $\tilde{v}^d_{i_1,i_2}$ in terms of a *single* order statistic only. The estimation will usually be more meaningful if we can exploit information provided by the gray level values of *all* the pixels in the prediction neighborhood set $\mathcal{N}_p$. In the next section, we investigate this possibility by extending the previous crisp estimation framework to a fuzzy estimation framework for $\tilde{v}^d_{i_1,i_2}$.

6.5 Fuzzy Prediction of Desired Gray Level Value

In this section, we define two fuzzy sets, namely the EDGE GRAY LEVEL ESTIMATOR (EG) fuzzy set and the TEXTURE GRAY LEVEL ESTIMATOR (TG) fuzzy set, over the domain of *gray level values* in an image. This fuzzy formulation is independent from our previous ETC-fuzzy formulation where the EDGE and TEXTURE fuzzy sets are defined over the domain of the *ETC measure*. The purpose of this second fuzzy formulation is to allow the utilization of different prediction strategies for $\tilde{v}^d_{i_1,i_2}$ in the edge and textured regions respectively, and the evaluation of this predicted gray level value using all the gray level values in the prediction neighborhood set $\mathcal{N}_p$ instead of a single crisp value.

6.5.1 Definition of the fuzzy estimator membership function

For this gray level estimator fuzzy model, we define the two set membership functions, $\varphi_{EG}(\tilde{v}_{i_1,i_2})$ and $\varphi_{TG}(\tilde{v}_{i_1,i_2})$ in terms of Gaussian functions, as opposed to the use of sigmoid nonlinearity in the first model. Again denoting the gray level value at location (i_1, i_2) as $\tilde{v}_{i_1,i_2}$ and the n_p-th order statistic of the prediction neighborhood set $\mathcal{N}_p$ as $\tilde{v}^{(n_p)}_{i_1,i_2}$, and assuming that $\tilde{v}_{i_1,i_2} \geq \overline{\tilde{v}}_{i_1,i_2}$ without loss of generality, which corresponds to the second condition in equations (6.26) and (6.27), we define the membership functions of the EG

fuzzy set and the TG fuzzy set as follows:

$$\varphi_{EG}(\widetilde{v}_{i_1,i_2}) = e^{-\xi_{EG}(\widetilde{v}_{i_1,i_2}-\widetilde{v}^{(6)}_{i_1,i_2})^2} \tag{6.28}$$

$$\varphi_{TG}(\widetilde{v}_{i_1,i_2}) = e^{-\xi_{TG}(\widetilde{v}_{i_1,i_2}-\widetilde{v}^{(7)}_{i_1,i_2})^2} \tag{6.29}$$

For the condition $\widetilde{v}_{i_1,i_2} < \overline{\widetilde{v}}_{i_1,i_2}$, we will replace the centers of the EG and TG membership functions with $\widetilde{v}^{(4)}_{i_1,i_2}$ and $\widetilde{v}^{(3)}_{i_1,i_2}$ respectively, in accordance with equations (6.26) and (6.27).

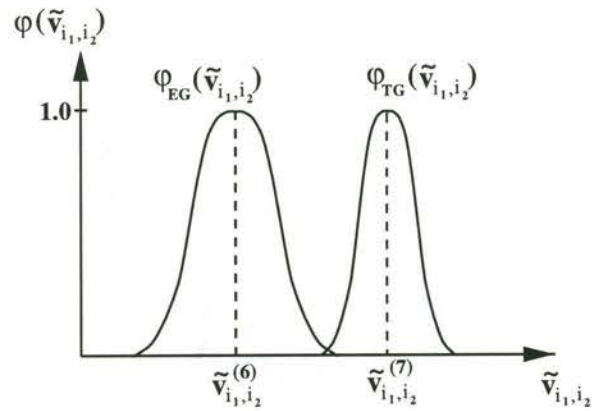


Figure 6.4: The fuzzy membership functions of the sets EG and TG

The two membership functions are depicted graphically in Figure 6.4. Under the condition $\widetilde{v}_{i_1,i_2} \geq \overline{\widetilde{v}}_{i_1,i_2}$, where we previously designate $\widetilde{v}^{(6)}_{i_1,i_2}$ as the preferred gray level estimator in the edge-oriented estimation scheme, we now generalize this concept by assigning a membership value of 1 for $\widetilde{v}^{(6)}_{i_1,i_2}$. However, instead of adopting this value exclusively, we have also assigned non-zero membership function values for gray level values close to $\widetilde{v}^{(6)}_{i_1,i_2}$, thus expressing the notion that nearby values also have relevancy in determining the final predicted gray level value. Similar assignments are adopted for the TG membership function with $\widetilde{v}^{(7)}_{i_1,i_2}$ as the center.

In the two equations, the two parameters ξ_{EG} and ξ_{TG} control the width of the respective membership functions, thus quantifying the notion of "closeness" in terms of gray level value differences for both fuzzy sets. In general, the values of these two parameters are different from each other, and the optimal values of both parameters may also differ for different image regions. In view of these, we have devised a training algorithm for adapting these parameters in different regions and this will be described in a later sec-

tion. The requirement for adaptation is also the reason we have chosen ξ_{EG} and ξ_{TG} to be multiplicative factors instead of their usual appearances as denominators in the form of variances for Gaussian functions: in the former case, we have only to ensure that both parameters are greater than zero. In the latter case, there is the additional difficulty that the adapting variances can easily become too small to result in an excessively large exponent for the Gaussian functions.

6.5.2 Fuzzy Inference Procedure for Predicted Gray Level Value

Given the membership functions $\varphi_{EG}(\widetilde{v}_{i_1,i_2})$ and $\varphi_{TG}(\widetilde{v}_{i_1,i_2})$ of the two fuzzy sets EG and TG, we can define a mapping F, which assigns a single fuzzy set G, the GRAY LEVEL ESTIMATOR fuzzy set, to each pair of fuzzy sets EG and TG,

$$G = F(EG, TG) \tag{6.30}$$

in the following way: since any fuzzy set is fully defined by specifying its membership function, we will define the set mapping F in terms of real-valued mappings between the parameters of the membership function $\varphi_G(\widetilde{v}_{i_1,i_2})$ and the parameters of the membership functions $\varphi_{EG}(\widetilde{v}_{i_1,i_2})$ and $\varphi_{TG}(\widetilde{v}_{i_1,i_2})$ as follows:

$$\varphi_G(\widetilde{v}_{i_1,i_2}) = e^{-\xi_G(\widetilde{v}_{i_1,i_2} - \widetilde{v}^G_{i_1,i_2})^2} \tag{6.31}$$

$$\widetilde{v}^G_{i_1,i_2} \equiv \widetilde{\mu}_E(\kappa_{i_1,i_2})\widetilde{v}^{(6)}_{i_1,i_2} + \widetilde{\mu}_T(\kappa_{i_1,i_2})\widetilde{v}^{(7)}_{i_1,i_2} \tag{6.32}$$

$$\xi_G \equiv \widetilde{\mu}_E(\kappa_{i_1,i_2})\xi_{EG} + \widetilde{\mu}_T(\kappa_{i_1,i_2})\xi_{TG} \tag{6.33}$$

under the condition $\widetilde{v}_{i_1,i_2} \geq \overline{\widetilde{v}}_{i_1,i_2}$. For $\widetilde{v}_{i_1,i_2} < \overline{\widetilde{v}}_{i_1,i_2}$, the terms $\widetilde{v}^{(6)}_{i_1,i_2}$ and $\widetilde{v}^{(7)}_{i_1,i_2}$ are replaced by $\widetilde{v}^{(4)}_{i_1,i_2}$ and $\widetilde{v}^{(3)}_{i_1,i_2}$ respectively.

The mapping operation is illustrated in Figure 6.5. From the figure, it is seen that the mapping performs a kind of interpolation between the two fuzzy sets EG and TG. The coefficients $\widetilde{\mu}_E(\kappa_{i_1,i_2})$ and $\widetilde{\mu}_T(\kappa_{i_1,i_2})$ refer to the previous normalized fuzzy coefficient values of the fuzzy ETC measure model, which is independent from the current gray level estimator fuzzy model. If the current pixel belongs to an edge, the conditions $\widetilde{\mu}_E(k) \approx 1$ and $\widetilde{\mu}_T(k) \approx 0$ are approximately satisfied, the resulting final gray level

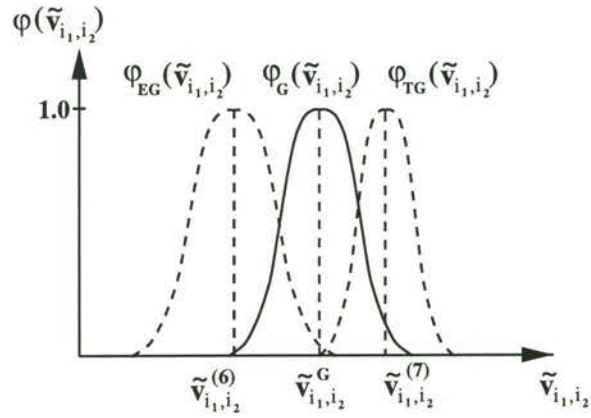


Figure 6.5: The mapping of the fuzzy sets EG and TG to G

estimator fuzzy set G is approximately equal to the edge gray level estimator fuzzy set EG, and the final prediction for the gray level value, $\tilde{v}^d_{i_1,i_2}$, will be based on the corresponding membership function of this fuzzy set. On the other hand, if the current pixel belongs to a textured area, the conditions $\tilde{\mu}_E(\kappa_{i_1,i_2}) \approx 0$ and $\tilde{\mu}_T(\kappa_{i_1,i_2}) \approx 1$ hold approximately, and the estimation of $\tilde{v}^d_{i_1,i_2}$ will be mostly based on the membership function of the texture gray level estimator fuzzy set TG. For all the other cases, the mapping operation results in a membership function for the set G with values of parameters intermediate between those of EG and TG.

6.5.3 Defuzzification of the fuzzy set G

The fuzzy inference procedure determines the final GRAY LEVEL ESTIMATOR fuzzy set G with corresponding membership function $\varphi_G(\tilde{v}_{i_1,i_2})$ from the membership functions $\varphi_{EG}(\tilde{v}_{i_1,i_2})$ and $\varphi_{TG}(\tilde{v}_{i_1,i_2})$. In order to provide a desired network output for the fuzzy HMBNN, we have to *defuzzify* the fuzzy set G to obtain a crisp prediction $\tilde{v}^d_{i_1,i_2}$ and the associated desired network output $\triangle\tilde{v}^d_{i_1,i_2}$. A common way to defuzzify a fuzzy set is to employ *centroid defuzzification* [61], where we assign the *centroid* of the fuzzy membership function to be the crisp value associated with the function. In the case of the fuzzy set G, we obtain

$$\tilde{v}^d_{i_1,i_2} \equiv \frac{\int_{-\infty}^{\infty} \tilde{v}\varphi_G(\tilde{v})d\,\tilde{v}}{\int_{-\infty}^{\infty} \varphi_G(\tilde{v})d\,\tilde{v}} = \tilde{v}^G_{i_1,i_2} \tag{6.34}$$

Due to the adoption of the Gaussian function for $\varphi_G(\tilde{v}_{i_1,i_2})$, this particular defuzzification strategy does not take into account the information provided by the width of the membership function. Specifically, rather than simply using $\tilde{v}^G_{i_1,i_2}$, which may not correspond to any of the pixel gray level values in the prediction neighborhood set, as an estimator for the current pixel value, we would also like to incorporate those pixels in the neighborhood set with values "close" to $\tilde{v}^G_{i_1,i_2}$, in a sense depending on the exact value of the width parameter ξ_G, in the determination of the final estimate for the current gray level value. The inclusion of these pixels from the neighborhood set will reduce incompatibility of the value of the estimator with those pixel values in its neighborhood. In accordance to these, we propose the following *discrete defuzzification procedure* for the fuzzy set G:

$$\tilde{v}^d_{i_1,i_2} \equiv \tilde{\varphi}_G(\tilde{v}^G_{i_1,i_2})\tilde{v}^G_{i_1,i_2} + \sum_{n_p=1}^{9} \tilde{\varphi}_G(\tilde{v}^{(n_p)}_{i_1,i_2})\tilde{v}^{(n_p)}_{i_1,i_2} \tag{6.35}$$

where $\tilde{\varphi}_G(\tilde{v}_{i_1,i_2})$ is the scaled membership function defined as follows:

$$\tilde{\varphi}_G(\tilde{v}_{i_1,i_2}) \equiv \frac{1}{C}\varphi_G(\tilde{v}_{i_1,i_2}) \tag{6.36}$$

$$C \equiv \varphi_G(\tilde{v}^G_{i_1,i_2}) + \sum_{n_p=1}^{9} \varphi_G(\tilde{v}^{(n_p)}_{i_1,i_2}) \tag{6.37}$$

From this expression, it is seen that the final crisp estimate $\tilde{v}^d_{i_1,i_2}$ does not necessarily equal to the centroid $\tilde{v}^G_{i_1,i_2}$ as in continuous defuzzification, but it also depends on the pixel values in the prediction neighborhood set $\mathcal{N}_p$. In particular, if there exists a pixel with value $\tilde{v}^{(n_p)}_{i_1,i_2}$ in $\mathcal{N}_p$ which is not exactly equal to $\tilde{v}^G_{i_1,i_2}$, but is nevertheless "close" to it in the form of a large membership value $\tilde{\varphi}_G(\tilde{v}^{(n_p)}_{i_1,i_2})$ in the fuzzy set G, then the crisp estimate $\tilde{v}^d_{i_1,i_2}$ will be substantially influenced by the presence of this pixel in $\mathcal{N}_p$ such that the final value will be a compromise between $\tilde{v}^G_{i_1,i_2}$ and $\tilde{v}^{(n_p)}_{i_1,i_2}$. This is to ensure that, apart from using $\tilde{v}^G_{i_1,i_2}$, we incorporate the information of all those pixels in the prediction neighborhood set which are close in values to $\tilde{v}^G_{i_1,i_2}$ to obtain a more reliable estimate of the current pixel value.

6.5.4 Regularization Parameter Update

Given the final crisp estimate $\tilde{v}^d_{i_1,i_2}$ for the current pixel, we can then define the desired network output $\triangle\tilde{v}^d_{i_1,i_2}$ for the fuzzy HMBNN as in Chapter 3:

$$\triangle\tilde{v}^d_{i_1,i_2} = \tilde{v}^d_{i_1,i_2} - \tilde{v}_{i_1,i_2} \tag{6.38}$$

Adopting the same cost function $\mathcal{E}_t$, where

$$\mathcal{E}_t = \frac{1}{2}(\triangle\tilde{v}^d_{i_1,i_2} - \triangle\tilde{v}_{i_1,i_2})^2 \tag{6.39}$$

we can update the value of the regularization parameter by applying stochastic gradient descent on this cost function:

$$\lambda_r(t+1) = \lambda_r(t) - \eta\frac{\partial\mathcal{E}_t}{\partial\lambda_r} \tag{6.40}$$

The partial derivative in the equation can be evaluated using the procedures described in Chapter 3.

However, unlike the case in Chapter 3, the fuzzy regularization network has two neurons, namely the texture neuron and the edge neuron, associated with a single sub-network for the combined edge/textured region, whereas there is only one neuron associated with a sub-network for our previous network. We must therefore derive two update equations for the two regularization parameters $\lambda^{tex}_{r_f}(t)$ and $\lambda^{edge}_{r_f}(t)$ associated with the respective neurons. In addition, each update equation is to be designed in such a way that the resulting value of the regularization parameter would be appropriate for the particular types of regions assigned to each neuron. For example, we may expect that the parameter $\lambda^{tex}_{r_f}(t)$ of the texture neuron would be smaller than the parameter $\lambda^{edge}_{r_f}(t)$ of the edge neuron due to the better noise masking capability of textures.

In view of these considerations, we can formulate the two update equations by regarding the current sub-network as a *two-layer network*. More precisely, we consider the form of equation (6.16), which is repeated here for convenience

$$\triangle\tilde{v}_{i_1,i_2} = \tilde{\mu}_E(\kappa_{i_1,i_2})\triangle\tilde{v}^{edge}_{i_1,i_2} + \tilde{\mu}_T(\kappa_{i_1,i_2})\triangle\tilde{v}^{tex}_{i_1,i_2}$$

where $\tilde{\mu}_E(\kappa_{i_1,i_2}), \tilde{\mu}_T(\kappa_{i_1,i_2})$ can be regarded as the *output weights* of the sub-network, and $\triangle\tilde{v}^{edge}_{i_1,i_2}, \triangle\tilde{v}^{tex}_{i_1,i_2}$ can be considered as the outputs of the hidden edge and texture neurons. As a result, we can derive the update equations for both $\lambda^{edge}_{r_f}$ and $\lambda^{tex}_{r_f}$ using the *generalized delta rule* for multilayer neural networks.

For the parameter $\lambda^{edge}_{r_f}$ associated with the edge neuron, the update equation is derived as follows:

$$\begin{aligned}
\lambda^{edge}_{r_f}(t+1) &= \lambda^{edge}_{r_f}(t) - \eta\frac{\partial\mathcal{E}_t}{\partial\lambda^{edge}_{r_f}} \\
&= \lambda^{edge}_{r_f}(t) - \eta\frac{\partial\mathcal{E}_t}{\partial\triangle\tilde{v}^{edge}_{i_1,i_2}}\frac{\partial\triangle\tilde{v}^{edge}_{i_1,i_2}}{\partial\lambda^{edge}_{r_f}} \\
&= \lambda^{edge}_{r_f}(t) + \eta(\triangle\tilde{v}^{d}_{i_1,i_2} - \triangle\tilde{v}_{i_1,i_2})\frac{\partial\triangle\tilde{v}_{i_1,i_2}}{\partial\triangle\tilde{v}^{edge}_{i_1,i_2}}\frac{\partial\triangle\tilde{v}^{edge}_{i_1,i_2}}{\partial\lambda^{edge}_{r_f}} \\
&= \lambda^{edge}_{r_f}(t) + \eta(\triangle\tilde{v}^{d}_{i_1,i_2} - \triangle\tilde{v}_{i_1,i_2})\tilde{\mu}_E(\kappa_{i_1,i_2})\frac{\partial\triangle\tilde{v}^{edge}_{i_1,i_2}}{\partial\lambda^{edge}_{r_f}} \qquad (6.41)
\end{aligned}$$

Similarly, for $\lambda^{tex}_{r_f}$, we have the following update equation

$$\lambda^{tex}_{r_f}(t+1) = \lambda^{tex}_{r_f}(t) + \eta(\triangle\tilde{v}^{d}_{i_1,i_2} - \triangle\tilde{v}_{i_1,i_2})\tilde{\mu}_T(\kappa_{i_1,i_2})\frac{\partial\triangle\tilde{v}^{tex}_{i_1,i_2}}{\partial\lambda^{tex}_{r_f}} \qquad (6.42)$$

From the equations, it is seen that if the current pixel belongs to a textured region, the conditions $\tilde{\mu}_T(\kappa_{i_1,i_2}) \approx 1$ and $\tilde{\mu}_E(\kappa_{i_1,i_2}) \approx 0$ is approximately satisfied, and we are essentially updating only the parameter of the texture neuron. On the other hand, if the pixel belongs to an edge, we have the conditions $\tilde{\mu}_E(\kappa_{i_1,i_2}) \approx 1$ and $\tilde{\mu}_T(\kappa_{i_1,i_2}) \approx 0$, and we are updating the edge neuron almost exclusively.

6.5.5 Update of the Estimator Fuzzy Set Width Parameters

Previously, it is seen that the width parameters of the gray level estimator fuzzy sets, ξ_{EG} and ξ_{TE} , determine to what degree the various pixels in the prediction neighborhood set participate in estimating the final predicted gray level value. In other words, the parameters establish the notion of "closeness" in gray level values for the various edge/textured regions, which may differ from one such region to another.

In general, we have no *a priori* knowledge of the values of the width parameters required for each combined edge/textured region. In view of this, we propose a simple

learning strategy which allows the parameters of each such region to be determined adaptively. Recall that, for the fuzzy sets EG and TG, we assign a membership value of 1 to the order-statistics $\tilde{v}^{(6)}_{i_1,i_2}$ and $\tilde{v}^{(7)}_{i_1,i_2}$ respectively (assuming $\tilde{v}_{i_1,i_2} > \overline{\tilde{v}}_{i_1,i_2}$), indicating that these two values are highly relevant in the estimation of the current gray level value. Since the shape of the Gaussian membership function can be fixed by two sample points, we can determine the width parameter by choosing a gray level value in the set of order statistics which is the *least* relevant in estimating the final gray level value, and assign a small membership value $\epsilon \ll 1$ to this gray level value, thus completely specifying the entire membership function.

A suitable candidate for this particular gray level value is the median, or alternatively, the 5-th order statistic $\tilde{v}^{(5)}_{i_1,i_2}$. If we adopt this value as our estimate for the final gray level value, we are effectively aiming at a median filtered version of the image. Although median filter is known to preserve edges while eliminating impulse noises for the case of images without blur [87] , we can expect that, for blurred or partially restored images, the median will be close to the mean gray level value in a local neighborhood, and thus would constitute an unsuitable estimate for the final gray level value in the vicinity of edges and textures. We can thus assign a small membership value to the median gray level value to completely determine the final form of the Gaussian membership function.

More precisely, we define the following error function $\mathcal{C}_\xi$ for adapting the two width parameters ξ_{EG} and ξ_{TG} as follows

$$\mathcal{C}_\xi = \frac{1}{2}(\epsilon - \varphi_G(\tilde{v}^{(5)}_{i_1,i_2}))^2 \tag{6.43}$$

which expresses the requirement that the value of $\varphi_G(\tilde{v}_{i_1,i_2})$ should be small when $\tilde{v}_{i_1,i_2}$ is close to the median value. The gradient descent update equations for the two parameters ξ_{EG} and ξ_{TG} with respect to $\mathcal{C}_\xi$ can then be derived by applying the chain rule according to equations (6.31) and (6.33). For the parameter ξ_{EG},

$$\begin{aligned}
\xi_{EG}(t+1) &= \xi_{EG}(t) - \eta_\xi \frac{\partial \mathcal{C}_\xi}{\partial \xi_{EG}} \\
&= \xi_{EG}(t) - \eta_\xi \frac{\partial \mathcal{C}_\xi}{\partial \varphi_G} \frac{\partial \varphi_G}{\partial \xi_G} \frac{\partial \xi_G}{\partial \xi_{EG}}
\end{aligned}$$

$$= \xi_{EG}(t) - \eta_\xi(-(\epsilon - \varphi_G(\tilde{v}^{(5)}_{i_1,i_2})))(-(\tilde{v}^{(5)}_{i_1,i_2} - \tilde{v}^{G}_{i_1,i_2})\varphi_G(\tilde{v}^{(5)}_{i_1,i_2}))(\tilde{\mu}_E(\kappa_{i_1,i_2}))$$

$$= \xi_{EG}(t) - \eta_\xi \tilde{\mu}_E(\kappa_{i_1,i_2})\varphi_G(\tilde{v}^{(5)}_{i_1,i_2})(\epsilon - \varphi_G(\tilde{v}^{(5)}_{i_1,i_2}))(\tilde{v}^{(5)}_{i_1,i_2} - \tilde{v}^{G}_{i_1,i_2}) \quad (6.44)$$

Similarly, the update equation for ξ_{TG} is

$$\xi_{TG}(t+1) = \xi_{TG}(t) - \eta_\xi \tilde{\mu}_T(\kappa_{i_1,i_2})\varphi_G(\tilde{v}^{(5)}_{i_1,i_2})(\epsilon - \varphi_G(\tilde{v}^{(5)}_{i_1,i_2}))(\tilde{v}^{(5)}_{i_1,i_2} - \tilde{v}^{G}_{i_1,i_2}) \quad (6.45)$$

From the form of the two update equations, it is seen that, similar to the case for the update of the regularization parameters, when the current pixel belongs to a textured region, where $\tilde{\mu}_T(\kappa_{i_1,i_2}) \approx 1$ and $\tilde{\mu}_E(\kappa_{i_1,i_2}) \approx 0$, substantial update is performed on ξ_{TG} while there is essentially no update on ξ_{EG}, which is reasonable in view of the necessity to update the shape of the TG membership function using information from the textured pixels only. On the other hand, for edge pixels where $\tilde{\mu}_E(\kappa_{i_1,i_2}) \approx 1$ and $\tilde{\mu}_T(\kappa_{i_1,i_2}) \approx 0$, only the parameter ξ_{EG} will be substantially modified.

6.5.6 Incorporation of the Texture Map

The incorporation of the above fuzzy models in the HMBNN enables the classification of the previous combined edge/textured regions into separate edge and textured regions, thus solving one of the stated problems of the previous version of the network in Chapter 3, namely the incompatibility of a single parameter value to both the edge and the textured regions. From the experimental results in Chapter 3, it was observed that whenever the adaptive regularization scheme was designed such that the resulting parameter values in the combined edge/textured regions are biased in favor of the textured areas, the edges will appear noisy. On the other hand, if the values are biased towards the edges, the textured regions will appear blurred. The capability of the current fuzzy HMBNN in distinguishing between the edge and texture components has allowed the possibility of adopting different regularization strategies for each of these regions, thus greatly alleviating this problem.

However, the other stated problem in Chapter 3, that of the appearance of smooth blotches within a textured area, remains unsolved by the current approach. As stated previously, this problem is due to the presence of regions of low local variances within

a textured area, which causes those regions to be classified as smooth regions in the segmentation step prior to restoration. Although there is justification in applying a large parameter value to those regions due to their low variances, the presence of such smooth regions within a textured area lends an unnatural appearance to the overall image. To solve this problem, we have to incorporate complementary information in the form of a texture map which represents the textured areas as continuous regions into the restoration process, in addition to the previous local variance-based segmentation.

In Chapter 5 we have derived a textured region extraction algorithm which approximately locates the most prominent textured areas in an image, and thus exactly serves the above purpose of providing an independent representation of the textured areas as continuous regions. This is possible due to the use of the ETC measure κ as an alternative characterization of the textures as opposed to local variances in the initial segmentation, and the incorporation of continuity constraints in deriving the final textured regions. For image restoration, we have to apply the texture extraction algorithm to the blurred and noisy images instead of the original image as in Chapter 5. For more accurate extraction results, we have first applied restoration with non-adaptive regularization to the respective images before texture extraction. The results of applying this algorithm to the degraded images of flower, eagle and Lena under different additive noise levels are shown in Figure 5.8 in Chapter 5.

With the availability of this additional characterization, we are especially interested in those pixels which are normally classified as smooth regions in the variance-based segmentation, but appear as within an extracted area in the current texture map. More precisely, denoting the set of extracted texture areas as $U_\tau^D, \tau = 1, \ldots, T$, we are interested in those pixels $(i_1, i_2) \in U_\tau^D \cap \mathcal{B}_{r_b}$ for some τ and r_b, where $\mathcal{B}_{r_b}$ is one of the smooth components in the variance-based segmentation. For these pixels, independent of the underlying value κ_{i_1,i_2} of the ETC measure, we *assign* the normalized fuzzy coefficient value of $\tilde{\mu}_T(\kappa_{i_1,i_2}) = 1$ and $\tilde{\mu}_E(\kappa_{i_1,i_2}) = 0$ to them, instead of determining the actual fuzzy membership values. This is due to the more correlated nature of the underlying gray level configuration of these *weak texture* pixels which usually leads to their identifi-

cation as edge elements instead of texture elements, and which in turn imposes a higher level of regularization to these pixels, again lending a smooth appearance to the weak textures. This can be compensated by adopting the above procedure of pre-assigning the membership function values to favor a texture classification for these pixels.

After identifying the weak texture pixels, we have to explicitly assign them to one of the combined edge/texture sub-networks for regularization, since they are excluded from the normal combined edge/textured regions in the variance-based segmentation map. For a particular weak texture pixel, we could have identified the closest combined edge/textured region in the variance-based segmentation map with respect to this pixel, and assign it to the corresponding sub-network. However, for simplicity, we simply assign all the weak texture pixels to a single combined edge/texture sub-network to avoid the need to evaluate distances for each of these pixels. A natural candidate for this single sub-network is the one with the area of its corresponding region being the maximum among all the regions. More precisely, we choose the sub-network with associated region $\mathcal{F}_{r_f^*}$ such that

$$r_f^* = \arg\max_{r_f} |\mathcal{F}_{r_f}| \tag{6.46}$$

where $|\mathcal{F}_{r_f}|$ denotes the area of region $\mathcal{F}_{r_f}$.

6.5.7 Experimental Results

As in Chapter 3, we have applied the fuzzy HMBNN regularization strategy to the images flower, Lena and eagle under various conditions of blur and additive noise, including 5 $\times$ 5 Gaussian blur ($\sigma_g = 1$) at 30dB BSNR, 5 $\times$ 5 uniform blur at 30dB BSNR, and 5 $\times$ 5 uniform blur at 20dB BSNR. (The original images of flower, Lena and eagle are shown in Figures 3.3(a), (b) and (c) in Chapter 3). In addition, we have provided a more comprehensive comparison between the performance of the current algorithm with other conventional algorithms by including the restoration results using Wiener filter [3] and the spatially adaptive iterative restoration algorithm by Kang and Katsaggelos [53, 56, 57]. In the latter algorithm, the regularization parameter was defined as a functional of the partially restored image $\hat{\mathbf{x}}$ at each iteration. In this way, estimation of the regularization

parameter value can proceed simultaneously with the restoration process.

In the spatially adaptive iterative restoration algorithm proposed by Kang and Katsaggelos [53, 56] (referred to as KK-SAI hereafter), a modified form of the original energy function for image restoration is defined as follows

$$E = \frac{1}{2}\|\mathbf{y} - \mathbf{H}\hat{\mathbf{x}}\|^2_{\mathbf{A}(\hat{\mathbf{x}})} + \frac{1}{2}\lambda(\hat{\mathbf{x}})\|\mathbf{D}\hat{\mathbf{x}}\|^2_{\mathbf{B}(\hat{\mathbf{x}})} \tag{6.47}$$

Instead of the usual isotropic Euclidean norm $\|\cdot\|^2$ employed previously, the weighted Euclidean norms with respect to the diagonal matrices $\mathbf{A}(\hat{\mathbf{x}})$ and $\mathbf{B}(\hat{\mathbf{x}})$ are adopted. These matrices allow the selective emphasis of either the data term or the regularization term in specific image pixels to achieve spatially adaptive image restoration. To achieve this purpose, they define the weighting matrix $\mathbf{B}(\hat{\mathbf{x}})$ as follows.

$$\mathbf{B}(\hat{\mathbf{x}}) = \mathbf{I} - \frac{\mathbf{\Theta}(\hat{\mathbf{x}})}{\gamma_{\hat{\mathbf{x}}}} \tag{6.48}$$

where

$$\mathbf{\Theta}(\hat{\mathbf{x}}) = diag[\sigma_1^2, \ldots\ldots, \sigma_{N_I}^2] \tag{6.49}$$

with σ_i^2 representing the local variance at the i-th pixel, and $\gamma_{\hat{\mathbf{x}}} = \max_i \sigma_i^2$. In high activity regions where the local variances are large, we should expect $b_{ij} \approx 0$, whereas $b_{ij} \approx 1$ in smooth regions.

The role of the weighting matrix $\mathbf{A}(\hat{\mathbf{x}})$ is complementary to that of $\mathbf{B}(\hat{\mathbf{x}})$ and is defined as follows:

$$\mathbf{A}(\hat{\mathbf{x}}) = \mathbf{I} - \mathbf{B}(\hat{\mathbf{x}}) \tag{6.50}$$

The global regularization parameter is defined as a function of the partially restored image $\hat{\mathbf{x}}$

$$\lambda(\hat{\mathbf{x}}) = \frac{\|\mathbf{y} - \mathbf{H}\hat{\mathbf{x}}\|^2_{\mathbf{A}(\hat{\mathbf{x}})}}{\frac{1}{\gamma_1} - \|\mathbf{D}\hat{\mathbf{x}}\|^2_{\mathbf{B}(\hat{\mathbf{x}})}} \tag{6.51}$$

where the constant γ_1 is chosen as follows

$$\gamma_1 = \frac{1}{2\|\mathbf{y}\|^2} \tag{6.52}$$

to ensure the overall convexity of the cost function with respect to $\hat{\mathbf{x}}$.

(a)

(b)

(c)

(d)

(e)

(f)

Figure 6.6: Restored flower images (5×5 Gaussian blur ($\sigma_g = 1$), 30dB BSNR). (a)-(f) Restored images using (a) Wiener filter (b) non-adaptive Hopfield restoration algorithm (optimally adjusted λ by user) (c) KK-SAI (d) HMBNN (texture-oriented prediction). (e) HMBNN (edge-oriented prediction) (f) Fuzzy HMBNN.

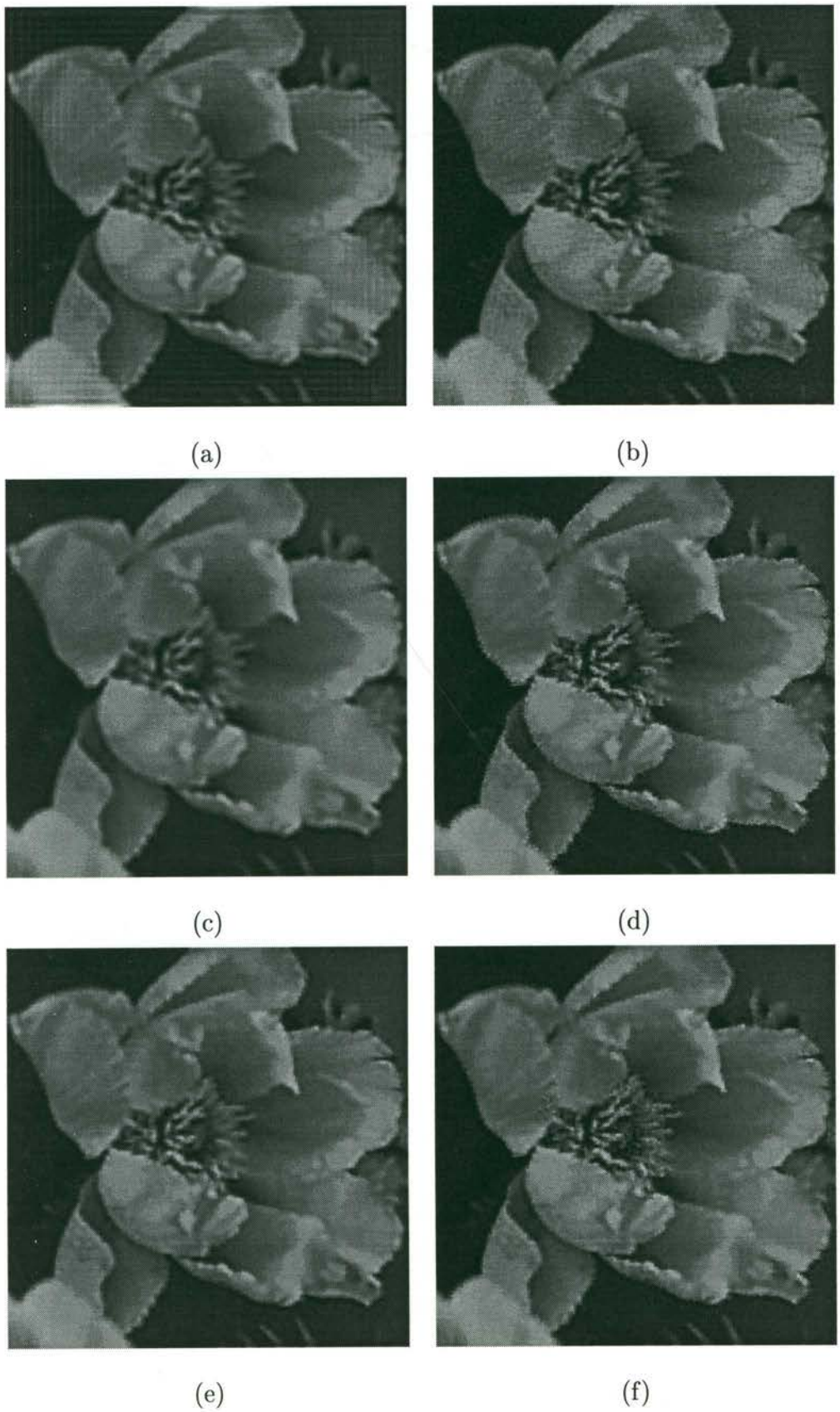


Figure 6.7: Restored flower images (5 × 5 uniform blur, 30dB BSNR). (a)-(f) Restored images using (a) Wiener filter (b) non-adaptive Hopfield restoration algorithm (optimally adjusted λ by user) (c) KK-SAI (d) HMBNN (texture-oriented prediction). (e) HMBNN (edge-oriented prediction) (f) Fuzzy HMBNN.

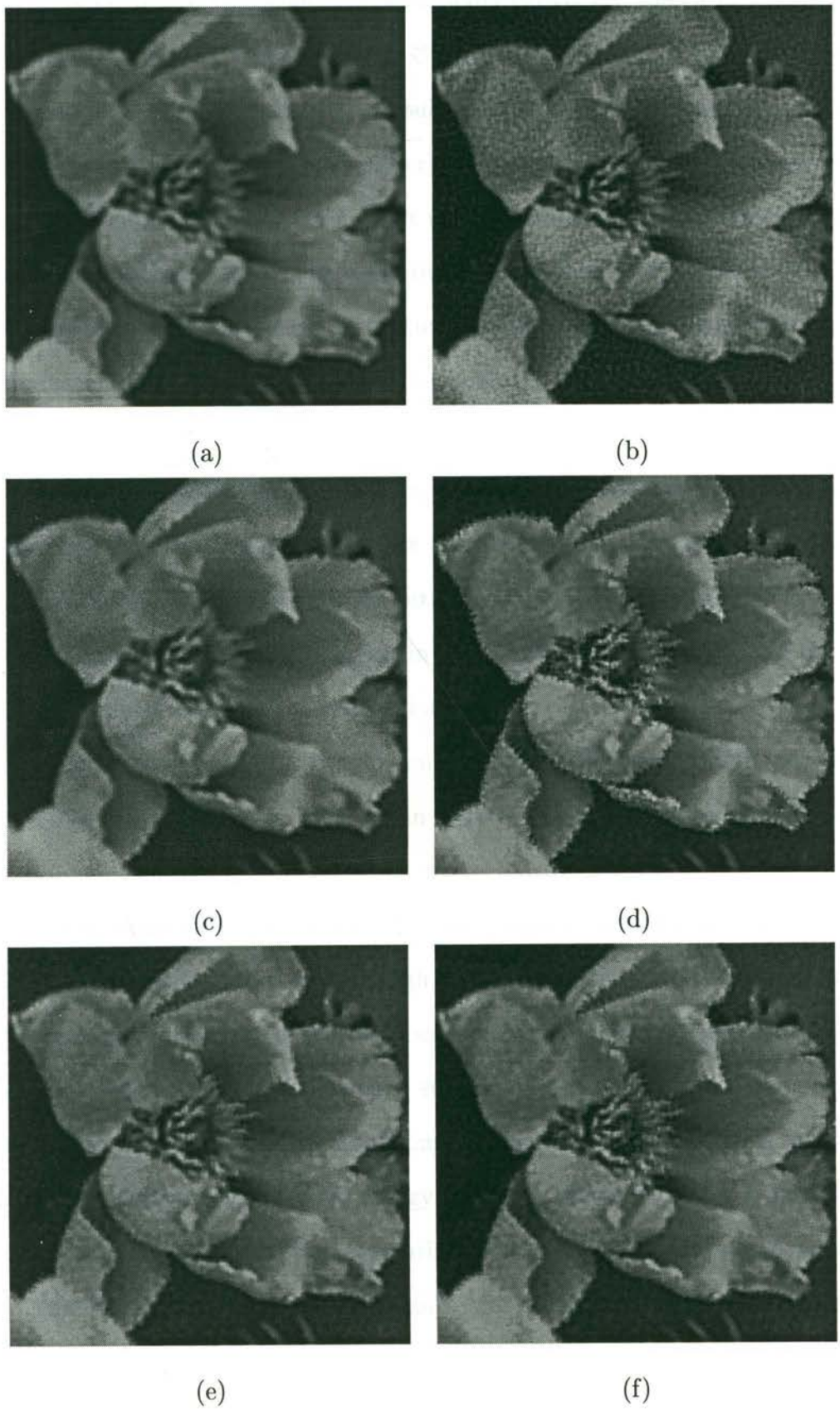


Figure 6.8: Restored flower image (5 × 5 uniform blur, 20dB BSNR). (a)-(f) Restored images using (a) Wiener filter (b) non-adaptive Hopfield restoration algorithm (optimally adjusted λ by user) (c) KK-SAI (d) HMBNN (texture-oriented prediction). (e) HMBNN (edge-oriented prediction) (f) Fuzzy HMBNN.

The results for 5 × 5 Gaussian blur with 30dB BSNR are shown in Figure 6.6 (the degraded image is shown in Figure 3.4(a) in Chapter 3). Figure 6.6(a) shows the result for Wiener filtering, Figure 6.6(b) shows the result using the Hopfield restoration network by Zhou *et.al* [118] (referred to as HNN hereafter), with the value of λ chosen to give the best visual results. Figure 6.6(c) shows the result using the KK-SAI restoration algorithm. In addition, we have included the results using the previous version of HMBNN in Chapter 3. Figure 6.6(d) shows the result using the texture-oriented prediction scheme for estimating the regularization parameters in the combined edge/textured regions, and Figure 6.6(e) shows the result using the edge-oriented prediction scheme. The result for the current fuzzy HMBNN algorithm is shown in Figure 6.6(f). Under this low level of degradation, we can observe that the restoration results are generally similar and comparable with each other. Although, compared with the fuzzy result in Figure 6.6(f), we can notice the slightly blurred image features and ringing in the Wiener filtering result, and the slightly blurred appearance of the KK-SAI restoration result.

The corresponding results for 5 × 5 uniform blur with 30dB BSNR are shown in Figure 6.7 (the degraded image is shown in Figure 3.5(a) in Chapter 3). In this case, the more ill-conditioned PSF causes more severe blurring of the image features in the Wiener filtering result in Figure 6.7(a). For the Hopfield network restoration result in Figure 6.7(b), it is evident that even though we have adjusted λ to give the best visual result, the non-adaptive nature of the process inevitably causes some blurring of the image features, while some residual noises still remain in the smooth regions. For the KK-SAI algorithm (Figure 6.7(c)), even though spatial adaptivity has been incorporated in the form of weighted norms in the energy function, the adoption of a fixed mapping between the local variance and local regularization parameter in the weighting matrices does not usually result in optimal visual qualities for different images under various forms of degradations. In the current case, although the strong edges and the smooth regions are properly regularized, the weak edges and textures appear blurred. For the non-fuzzy version of the HMBNN restoration algorithm, although the result using the texture-oriented prediction scheme in Figure 6.7(d) provides a satisfactory result compared with

the previous algorithms, where we can notice the comparative absence of background noise and the relatively more well-defined flower stamen, we can also notice the appearance of noises in the vicinity of edges. The noise is less noticeable in the textured areas such as the flower stamen area due to their better noise masking capability. If we try to suppress the noises around the edges by using the edge-oriented prediction scheme (Figure 6.7(e)), we can notice that, although the edges in the image are satisfactorily restored, the textured areas now become blurred due to the higher values of λ required for noise smoothing around the edges. These can be compared with the restoration result using the current fuzzy algorithm in Figure 6.7(f), where different values of λ are applied to the textured areas and edges respectively according to the local ETC measure value. We can observe that noise suppression around the edges is achieved without compromising the details of the textured area, resulting in a very satisfactory quality for the final restored image.

The necessity for the current fuzzy algorithm becomes more apparent if we look at the results under 5×5 uniform blur with 20dB BSNR, representing the most severe degradation in this experimental setup (the degraded image is shown in Figure 3.6(a) in Chapter 3). We can notice the blurry appearance of the Wiener filtering result in Figure 6.8(a), and the noisy appearances of the Hopfield network result by Zhou *et al* in Figure 6.8(b). Similar to the 30dB case, the KK-SAI algorithm results in properly regularized smooth regions and strong edges, while the weak edges and textures remain blurred. Although the restoration using the non-fuzzy HMBNN and texture-oriented prediction in Figure 6.8(d) can be considered satisfactory, the noises around the edges are more noticeable than for the 30dB BSNR case. If we apply edge-oriented prediction to suppress the noises around the edges, the blurring of the textured areas becomes apparent (Figure 6.8(e)).

These restored images can be compared with the fuzzy restoration result in Figure 6.8(f). we can see that the restoration result is satisfactory even under this high level of degradation. This is achieved by the possibility of distinguishing between textures and edges through the ETC measure and the judicious assignment of different λ values to the two feature classes.

(a)

(b)

(c)

(d)

(e)

(f)

Figure 6.9: Restored Lena images (5×5 uniform blur, 30dB BSNR). (a)-(f) Restored images using (a) Wiener filter (b) non-adaptive Hopfield restoration algorithm (optimally adjusted λ by user) (c) KK-SAI (d) HMBNN (texture-oriented prediction). (e) HMBNN (edge-oriented prediction) (f) Fuzzy HMBNN.

(a)

(b)

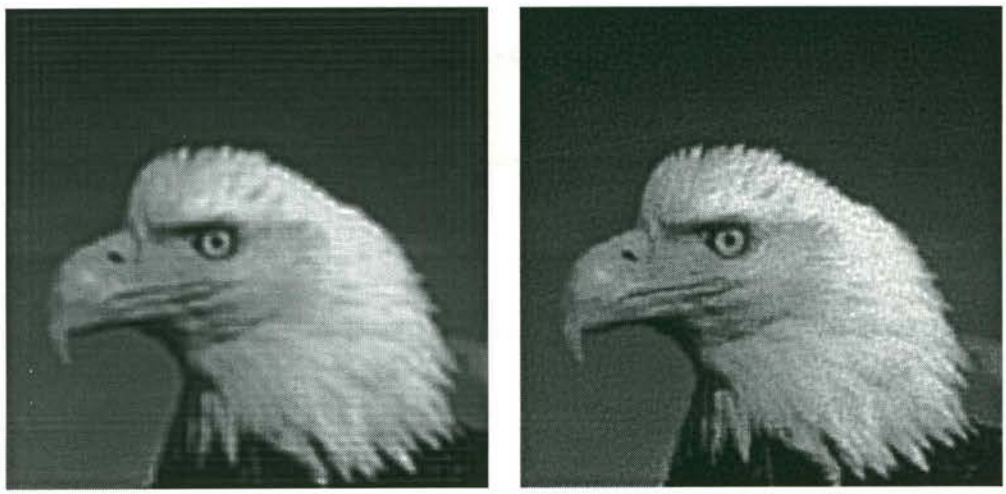
(c)

(d)

(e)

(f)

Figure 6.10: Restored Lena images (5×5 uniform blur, 20dB BSNR). (a)-(f) Restored images using (a) Wiener filter (b) non-adaptive Hopfield restoration algorithm (optimally adjusted λ by user) (c) KK-SAI (d) HMBNN (texture-oriented prediction). (e) HMBNN (edge-oriented prediction) (f) Fuzzy HMBNN.



(a)

(b)

(c)

(d)

(e)

(f)

Figure 6.11: Restored eagle images (5×5 uniform blur, 30dB BSNR). (a)-(f) Restored images using (a) Wiener filter (b) non-adaptive Hopfield restoration algorithm (optimally adjusted λ by user) (c) KK-SAI (d) HMBNN (texture-oriented prediction). (e) HMBNN (edge-oriented prediction) (f) Fuzzy HMBNN.

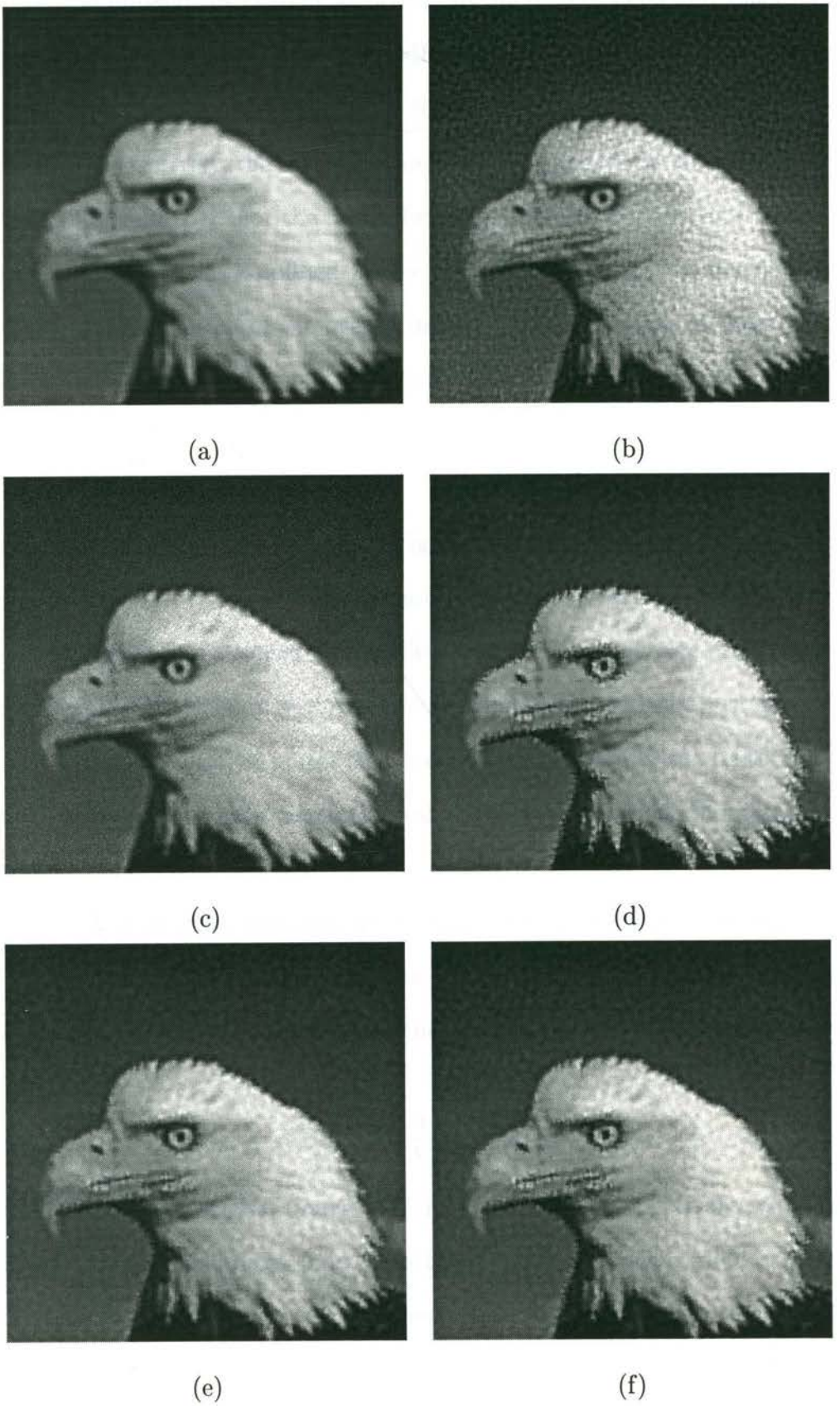


Figure 6.12: Restored eagle images (5×5 uniform blur, 20dB BSNR). (a)-(f) Restored images using (a) Wiener filter (b) non-adaptive Hopfield restoration algorithm (optimally adjusted λ by user) (c) KK-SAI (d) HMBNN (texture-oriented prediction). (e) HMBNN (edge-oriented prediction) (f) Fuzzy HMBNN.

We have also applied the current algorithm to the images Lena and Eagle, under 5 × 5 uniform blur at both 30dB BSNR and 20dB BSNR, in Figures 6.9, 6.10, 6.11 and 6.12 (the degraded images of Lena and eagle at 30dB BSNR are shown in Figures 3.9(a) and Figures 3.10(a) respectively, the corresponding degraded images at 20dB BSNR are shown in Figures 3.12(a) and (b)). For the image Lena at 20dB BSNR in Figure 6.10 , it is seen in Figure 6.10(d) that the amplified noises around those edges on the face especially interfere with the perception of the overall image. The suppression of those noises using edge-oriented prediction results in a more noticeable blurring due to the large textured areas in the image. The fuzzy restoration result in Figure 6.10(f) alleviates these two problems to a large extent.

For the eagle image at 20dB BSNR in Figure 6.12, the primary purpose of employing the fuzzy HMBNN is for edge noise suppression (compare Figures 6.12(d) and (f)). Due to the comparative lack of strongly textured regions, the result of edge-oriented prediction using the non-fuzzy HMBNN in Figure 6.12(e) is comparable with the current fuzzy result in Figure 6.12(f). However, we notice slight noise amplification in the textures below the eagle's eye. This is due to the particular value of $\lambda_{r_f}^{tex}$ selected by the algorithm under the assumption that textural patterns can effectively mask noise, which is not strictly satisfied in the current case, although the overall effect of this is not too objectionable.

In addition to the subjective comparisons, we have included objective comparisons of the restoration results in terms of the root mean square error (RMSE) measure, which is defined as follows:

$$RMSE = (\frac{1}{N_I}\|\mathbf{x} - \hat{\mathbf{x}}\|^2)^{\frac{1}{2}} \tag{6.53}$$

where $\mathbf{x}$ and $\hat{\mathbf{x}}$ are the original image and restored image respectively with the pixels arranged in lexicographical order, and N_I is the number of pixels in the image.

The RMSE values of the restored images using the various algorithms are listed in Table 6.1 . It can be seen that the current HMBNN approach results in the lowest RMSE for all the images under various forms of degradations. However, we should bear in mind that the RMSE measure just represents a convenient supplementary reference, and does

Image, PSF/noise level	Blurred	Wiener	HNN	KK-SAI	Fuzzy HMBNN
Flower, Uniform PSF/30dB	8.73	7.96	5.76	6.95	5.16
Flower, Uniform PSF/20dB	10.06	8.12	8.71	8.69	6.94
Lena, Uniform PSF/30dB	13.03	11.23	8.18	11.15	7.10
Lena, Uniform PSF/20dB	13.79	11.61	10.67	12.05	9.69
Eagle, Uniform PSF/30dB	9.88	9.61	7.92	8.64	6.97
Eagle, Uniform PSF/20dB	11.98	9.75	10.56	11.12	8.99

Table 6.1: RMSE values of the restoration results using various algorithms

not fully characterize the visual quality of an image. This can be seen by comparing the RMSE values of the Kang and Katsaggelos spatially adaptive iterative algorithm with that of the Wiener filter under 20dB noise. Although the RMSE values of the Wiener filtering result are lower than that of the spatially adaptive algorithm at this noise level, it is obvious upon visual inspection that the spatially adaptive algorithm provides a more acceptable restoration result due to the comparative absence of ringing in the background.

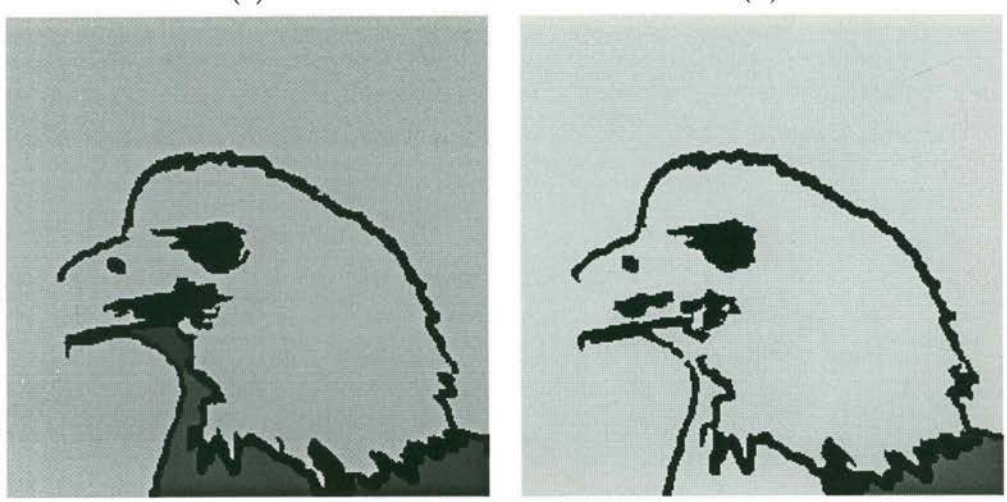
The λ-maps for the three images are shown in Figure 6.13. The maps corresponding to 5×5 times uniform blur at 30dB BSNR for the images are shown in Figures 6.13(a),(c),(e), while those corresponding to the same PSF at 20dB BSNR are shown in Figures 6.13(b),(d),(f). Since two regularization parameters, namely $\lambda_{r_f}^{edge}$ and $\lambda_{r_f}^{tex}$, are defined for each combined edge/textured region and applied to each pixel within, we have used the convex combination (with respect the normalized fuzzy coefficients) of these two values at each pixel for the purpose of constructing the λ-map.

$$\lambda_{r_f} = \tilde{\mu}_E(\kappa_{i_1,i_2})\lambda_{r_f}^{edge} + \tilde{\mu}_T(\kappa_{i_1,i_2})\lambda_{r_f}^{tex} \tag{6.54}$$

From the figures, it is seen that the maps are in general similar to those produced by the non-fuzzy HMBNN restoration algorithm, where there are tendencies for the designated smooth regions to assume larger areas at higher noise levels. However, a distinguishing difference between the current λ-maps and those in Chapter 3 is the explicit superposition

(a) (b)

(c) (d)



(e) (f)

Figure 6.13: The λ-distribution maps for the three images under 5 $\times$5 uniform blur at different levels of additive noise. (a)-(b) Flower (a) 30dB BSNR (b) 20dB BSNR (c)-(d) Lena (c) 30dB BSNR (d) 20dB BSNR (e)-(f) Eagle (e) 30dB BSNR (f) 20dB BSNR.

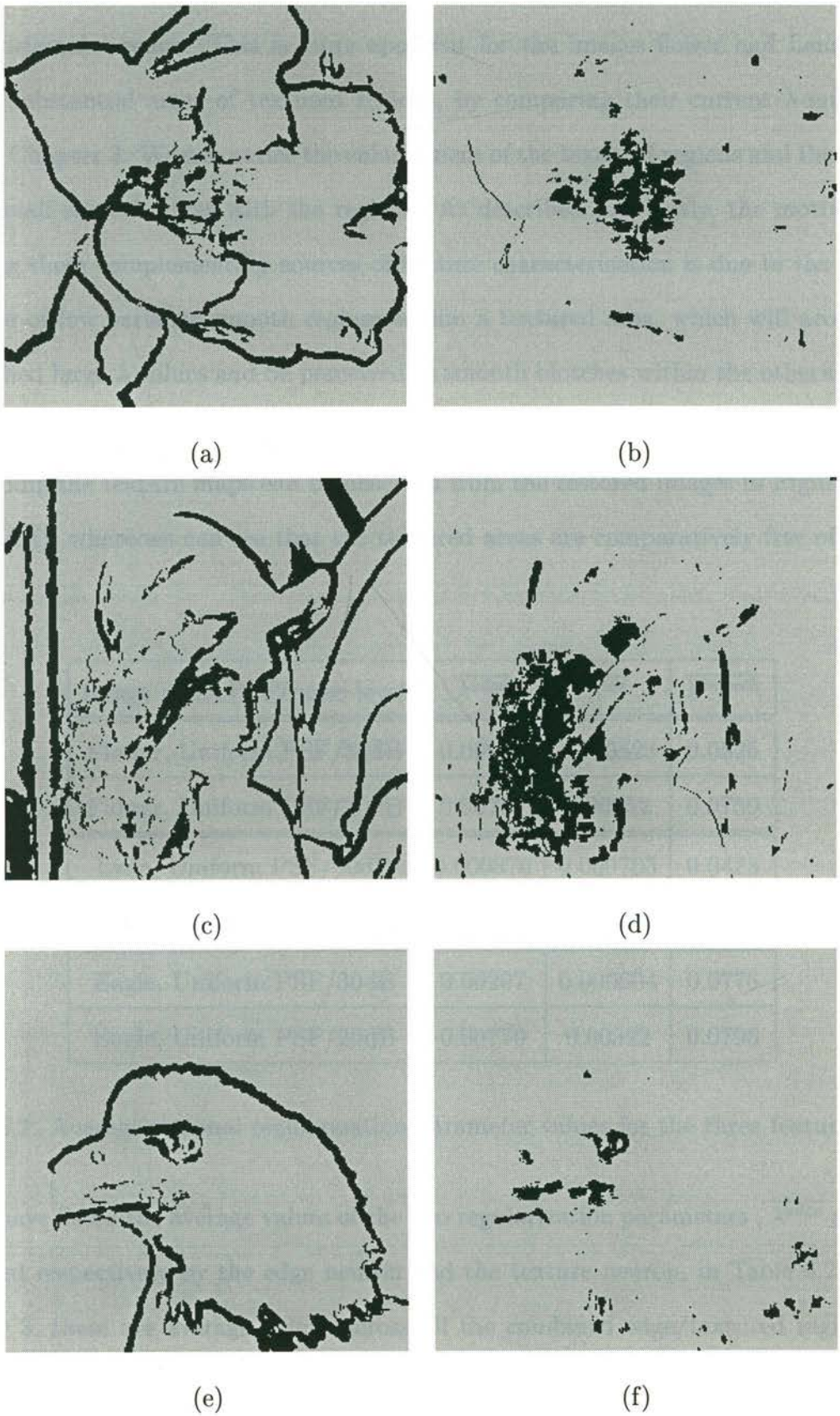


Figure 6.14: (a)-(b) Flower (a) edge pixels. (b) texture pixels (c)-(d) Lena (c) edge pixels (d) texture pixels. (e)-(f) Eagle (e) edge pixels (f) texture pixels.

of the derived texture map, using the texture extraction algorithm described in Chapter 5, on the original λ-maps. This is more apparent for the images flower and Lena, which contain substantial areas of textured regions, by comparing their current λ-maps with those in Chapter 3. We can notice the enlargement of the textured regions and the removal of the small smooth areas with the regions. As described previously, the motivation of including these complementary sources of texture characterization is due to the possible existence of low variance smooth regions within a textured area, which will accordingly be assigned large λ values and be perceived as smooth blotches within the otherwise well-defined textural patterns, giving an unnatural appearance to the overall image. The effect of including the texture maps can be observed from the restored images in Figures 6.8(f) and 6.10(f), where we can see that the textured areas are comparatively free of smooth blotches.

Image name,PSF/noise level	$\overline{\lambda^{edge}}$	$\overline{\lambda^{tex}}$	$\overline{\lambda^{smooth}}$
Flower, Uniform PSF/30dB	0.00156	0.000523	0.0525
Flower, Uniform PSF/20dB	0.00767	0.00252	0.0759
Lena, Uniform PSF/30dB	0.000976	0.000763	0.0423
Lena, Uniform PSF/20dB	0.00568	0.00238	0.0690
Eagle, Uniform PSF/30dB	0.00207	0.000604	0.0776
Eagle, Uniform PSF/20dB	0.00770	0.00322	0.0795

Table 6.2: Average regional regularization parameter values for the three feature types

We have listed the average values of the two regularization parameters , $\overline{\lambda^{edge}}$ and $\overline{\lambda^{tex}}$, estimated respectively by the edge neuron and the texture neuron, in Table 6.2 . As in Chapter 3, these are average values across all the combined edge/textured regions and weighted by the region areas. For the 30dB BSNR cases, it can be observed that due to the different training strategies applied to the edge neuron and texture neuron, the values of the corresponding parameters exhibit marked differences. Most notably, the texture

parameters $\overline{\lambda^{tex}}$ are consistently smaller than the edge parameters. This is appropriate for the enhancement of the textured areas without causing too noticeable distortions due to their better noise masking capability. On the other hand, the comparatively larger edge parameters $\overline{\lambda^{edge}}$ allow the smoothing of the more noticeable noises in the vicinity of edges. However, compared with the smooth region parameter values $\overline{\lambda^{smooth}}$, the edge parameters are still small enough to provide an acceptable enhancement of the edges. For the 20dB BSNR cases, similar conclusions can be drawn, except that all the corresponding parameter values are greater than the 30dB BSNR cases in order to suppress the higher levels of noises. In addition, the differences between $\overline{\lambda^{edge}}$ and $\overline{\lambda^{tex}}$ values become greater due to the need to suppress the even more noticeable noises around the edges.

Employing the fuzzy inference mechanism to determine the desired network output $\triangle\tilde{v}^d_{i_1,i_2}$ at each pixel, we could expect that this will vary from pixel to pixel in such a way to allow the preferential enhancement of the textured areas to edges. This cannot be readily observed from the λ-maps since we are performing the fuzzy inference on the gray level update values instead of the λ values themselves. To appreciate this adaptivity more fully, recall that for those pixels with their ETC fuzzy coefficient values $\tilde{\mu}_T(\kappa_{i_1,i_2}) \gg \tilde{\mu}_E(\kappa_{i_1,i_2})$, they will be essentially classified as textures and assigned the texture regularization parameter $\lambda^{tex}_{r_f}$. On the other hand, for those pixels with coefficient values $\tilde{\mu}_E(\kappa_{i_1,i_2}) \gg \tilde{\mu}_T(\kappa_{i_1,i_2})$, they will essentially be considered as edge points and assigned the edge regularization parameter $\lambda^{edge}_{r_f}$. In view of these, we have displayed, in Figures 6.14(b), (d) and (f), those pixels with their ETC coefficient values $\tilde{\mu}_T(\kappa_{i_1,i_2}) > \tilde{\mu}_E(\kappa_{i_1,i_2})$. We should expect that those pixels will be regularized essentially by the smaller $\lambda^{tex}_{r_f}$, therefore it is important that they actually correspond to the textured areas in the image. This can be confirmed from the figures where it is seen that the displayed areas in the Lena image are mostly clustered around the feathers, and those in the flower image appears within the area containing the stamen. In Figures 6.14(a), (c) and (e), we have displayed those image pixels with their ETC coefficient values $\tilde{\mu}_E(\kappa_{i_1,i_2}) > \tilde{\mu}_T(\kappa_{i_1,i_2})$. They will essentially be regularized with the larger parameter $\lambda^{edge}_{r_f}$ and thus we expect that they should correspond to the edges of the image, which

is again confirmed from the figures.

With the availability of this edge-texture discrimination information, together with the auxiliary texture map, further enhancement of the current algorithm is possible: our current image model assumes that the texture patterns are capable of effectively masking the noises. If the noise level or the specific texture patterns concerned are such that this assumption is not valid, we can always incorporate the associated specific noise or texture model in deriving a more appropriate local regularization scheme. It is then possible to restrict its application to the textured region due to the availability of the auxiliary texture information, and leave the other regions unaffected.

6.6 Summary

We have generalized the previous HMBNN framework for adaptive regularization to incorporate fuzzy information for edge/texture discrimination. Adopting the ETC measure developed in Chapter 5, we propose an edge/texture fuzzy model which expresses the degree to which a local image neighborhood resembles either edge or texture in terms of two fuzzy membership values. Correspondingly, we modify our previous network architecture to incorporate two neurons, namely the edge neuron and the texture neuron, within each sub-network. Each of the two neurons estimate an independent regularization parameter from the local image neighborhood and evaluate the corresponding required gray level update value as a function of its own parameter. The two gray level update values are then combined using fuzzy inference to produce the final required update in such a way which takes into account the degree of edge/texture resemblance of the local neighborhood. In general, the algorithm is designed such that less regularization is applied to the textured areas due to their better noise masking capability, and more regularization is applied to the edges where the noises are comparatively more visible.

The generalized algorithm is applied to a number of images under various conditions of degradations. The better visual quality of the restored images under the current algorithm can be appreciated by comparing the result with those produced using a number

of conventional restoration algorithms, and especially those using the previous non-fuzzy version of the current algorithm where a single regularization parameter value is applied to both textured areas and edges whenever they are connected as a single region. The current fuzzy HMBNN paradigm thus refines the previous notion of simply applying less regularization for the combined edge/textured regions to allow the possibility of using different levels of regularization to accommodate the different noise masking capabilities of the various regional components, which represents a novel contribution to the field of adaptive image regularization.

This Chapter concludes our work on adaptive regularization using HMBNN. In the next Chapter, we will adopt a completely new viewpoint for the adaptive regularization problem in terms of a novel cost measure which governs the adaptive assignment of the regularization parameters to various image regions. Due to the difficulty of optimizing this cost measure using conventional gradient-based algorithms, we will introduce the application of evolutionary computation to the problem of adaptive regularization.

Chapter 7

Application of Evolutionary Programming to Adaptive Regularization

7.1 Introduction

In this Chapter, we propose an alternative solution to the problem of adaptive regularization by adopting a new *global* cost measure. It was seen in Chapters 5 and 6 that the newly formulated ETC measure is capable of distinguishing between the textures and edges in an image. It is further observed in this Chapter that the distribution function of this measure value in a typical image assumes a characteristic shape, which is comparatively invariant across a large class of images, and can thus be considered a *signature* for the images. This is in contrast with the distribution function of other quantities such as the gray level values which varies widely from image to image, as can be confirmed by observing the gray level histograms of different images.

It is also observed that the corresponding ETC distribution function (or equivalently the ETC probability density function (ETC-pdf)) of degraded images is usually very different from that of the non-degraded images. In other words, for optimal restoration results, we can assign the regularization parameters in such a way that the ETC distri-

bution function of the restored image once again assumes the shape which is typical for non-degraded images, or more formally, we have to minimize the distance between the ETC-pdf of the restored image and the characteristic ETC-pdf of non-degraded images.

In practice, we can only approximate the ETC-pdf by the histogram of the measure values in the image, which cannot be expressed in closed form with respect to the regularization parameters, and thus the conventional gradient-based optimization algorithms are not applicable. We have therefore adopted an artificial evolutionary optimization approach where evolutionary programming (EP), belonging to the class of algorithms known as evolutionary computational algorithms, is used to search for the optimal set of regularization parameters with respect to the ETC-pdf criterion. One of the advantages of this class of algorithms is their independence from the availability of gradient information, which is therefore uniquely suited to our current optimization problem. In addition, these algorithms employ multiple search points in a population instead of a sequence of single search points as in conventional optimization algorithms, thus allowing many regions of the parameter space to be explored simultaneously. The characteristics of this class of algorithms are described in the following section.

7.2 Introduction to Evolutionary Computation

Evolutionary programming [29, 30] belongs to the class of optimization algorithms known as evolutionary computational algorithms [7, 8, 9, 10, 29] which mimic the process of natural evolution to search for an optimizer of a cost function. There are three mainstreams of research activities in this field, which include research in genetic algorithm (GA) introduced by Holland [35, 43, 75] , evolutionary programming (EP) by Fogel [29, 30] , and evolutionary strategy (ES) by Rechenberg and Schwefel [95, 96]. The defining characteristic of this class of algorithms includes its maintenance of a diversity of potential optimizers in a *population* and allow highly effective optimizers to emerge through the processes of mutation, recombination, competition and selection. The implicit parallelism resulting from the use of multiple search points instead of a single search point in conventional

optimization algorithms allows many regions of the search space to be explored simultaneously. Together with the stochastic nature of the algorithms which allow search points to spontaneously escape from non-global optima, and the independence of the optimization process from gradient information, the instances of local minima are usually reduced, as unlike the case with the gradient-based algorithms. In addition, this independence from gradient information allows the incorporation of highly irregular functions as fitness criteria for the evolutionary process, unlike the case with gradient-based algorithms where only differentiable cost functions are allowed. It is not even necessary for the cost function to have a closed form, as in those cases where the function values are obtained only through simulation. We will give a brief introduction of the above members of this class of algorithms in the following sections.

7.2.1 Genetic Algorithm (GA)

Genetic algorithm [35, 43, 75] is the most widely used among the three evolutionary computational algorithms. The distinguishing feature of GA includes its representation of the potential optimizers in the population as *binary strings*. Assuming the original optimizers are real-valued vectors $\mathbf{z} \in \mathbf{R}^N$, an encoding operation $\mathcal{G}$ is applied to each of these vectors to form the binary strings $\mathbf{g} = \mathcal{G}(\mathbf{z}) \in \mathbf{B}^H$, where $\mathbf{B} = \{0, 1\}$. In other words, the various evolutionary operations are carried out in the space of *genotypes*.

New individuals in the population are created using the operations of *crossover* and *mutation*. In GA, crossover is the predominant operation, while mutation only serves as an infrequent background operation. In crossover, two binary strings $\mathbf{g}_{p_1}, \mathbf{g}_{p_2}$ are randomly selected from the population. A random position $h \in \{1, \ldots, H\}$ is selected along the length of the binary string. After this, the sub-string to the left of h in p_1 is joined to the right sub-string of p_2, and similarly for the left sub-string of p_2 and the right sub-string of p_1, thus mimicking the biological crossover operation on the chromosomes. On the other hand, the mutation operator toggles the status of each bit for a certain binary string with a probability π_m, which is a very small value in the case of GA. The main purpose of

mutation is to introduce new variants of genotypes into the population.

After the processes of crossover and mutation, the fitness of each binary string, $f(\mathbf{g}_p), p = 1, \ldots, \mu$, is evaluated, where μ is the number of optimizers in the population, and f represents the *fitness function*. The function f usually reflects the requirement of the optimization problem at hand. In the case of a maximization task, we can usually equate the objective function of the problem with the fitness function. In the case of a problem which requires minimization, we can simply set the fitness function to be the negative of the current cost function.

After the evaluation of the individual fitness values, the optimizers in the population undergo a proportional selection process: each optimizer is to be included in the next generation with probability $\pi(\mathbf{g}_{p'})$, which reflects the relative fitness of individual $\mathbf{g}_{p'}$

$$\pi(\mathbf{g}_{p'}) = \frac{f(\mathbf{g}_{p'})}{\sum_{p=1}^{\mu} f(\mathbf{g}_p)} \tag{7.1}$$

In other words, an individual is more likely to survive into the next generation if it possesses a high fitness value. As the algorithm proceeds, the population will eventually consist of those optimizers with appreciable fitness values.

7.2.2 Evolutionary Strategy (ES)

As opposed to genetic algorithm, evolutionary strategy [95, 96] represents an alternative evolutionary approach where the various adaptation operations are carried out in the space of *phenotypes*: instead of first encoding the individual optimizers $\mathbf{z} \in \mathbf{R}^N$ into binary strings, the *recombination* and *mutation* operations are carried out directly on the real-valued vectors as described below.

For the recombination operation, two optimizers $\mathbf{z}_{p_1}, \mathbf{z}_{p_2}$ are randomly selected from the population, and a new optimizer $\mathbf{z}$ is generated from these two according to the recombination operator C.

$$\mathbf{z} = C(\mathbf{z}_{p_1}, \mathbf{z}_{p_2}) \tag{7.2}$$

The simplest form for C is the linear combination operation

$$\mathbf{z} = \alpha \mathbf{z}_{p_1} + (1 - \alpha)\mathbf{z}_{p_2} \tag{7.3}$$

where $\alpha < 1$ is the combination coefficient, although other combination operations, as described in [7], are also possible.

The mutation operation in ES randomly perturbs each component of the optimizer vector z_j to form a new optimizer $\mathbf{z}'$ with components z'_j as follows

$$z'_j = z_j + N(0, \sigma_j^m) \tag{7.4}$$

where $N(0, \sigma)$ is a Gaussian random variable with mean 0 and standard deviation σ. In the terminology of ES, the parameter σ_j^m associated with the component z_j is usually referred to as a mutation *strategy parameter*. The values of the mutation strategy parameters determine whether the current optimization process more resembles a global search, as when the σ_j^m's assume large values, which is desirable at the initial stages when many regions of the parameter space are simultaneously explored, or a local search which is more appropriate toward the final stages when the mutation strategy parameters assume small values to restrict the search within promising localities. The mutation strategy parameters themselves are usually adapted according to the following log-normal equation:

$$\sigma_j'^m = \sigma_j^m \exp(\tau' N(0,1) + \tau N_j(0,1)) \tag{7.5}$$

where τ', τ are pre-determined constants, and $N(0,1), N_j(0,1)$ are Gaussian random variables with mean 0 and standard deviation 1. In the case of $N_j(0,1)$, the random variable is re-sampled for each new component j. The log-normal adaptation equation is adopted to preserve the positivity of the mutation strategy parameters σ_j^m.

The recombination and mutation operations are performed γ times to form γ new individuals from the original μ individuals in the population. In the case of $(\mu+\gamma)$ selection strategy [7] , the fitness values $f(\mathbf{z}_p)$ of each individual in the $(\mu + \gamma)$ parent/descendant combination are evaluated, and those μ optimizers in this combination with the greatest fitness values are incorporated into the population in the next generation. In the (μ, γ) selection strategy [7], only the fitness values of the newly generated γ descendants are evaluated and the fittest of those incorporated into the next generation. The selection process is deterministic and depends solely on the fitness values of the individuals.

7.2.3 Evolutionary Programming (EP)

Evolutionary Programming [29, 30] shares many common features with evolutionary strategy in that the primary adaptation operations are also carried out in the space of phenotypes. In addition, there are a number of important similarities.

- The individual components z_j of each optimizer $\mathbf{z} \in \mathbf{R}^N$ are also perturbed according to equation (7.4), with the mutation strategy parameters σ_j^m similarly defined for the current component. In some variants of EP, the individual components are perturbed by an amount proportional to the square root of the objective function value [29].

- The individual mutation strategy parameters σ_j^m themselves are also subject to adaptations. In some EP variants, the log-normal relationship (7.5) is also adopted for this purpose.

Despite these similarities, there are a number of important differences between EP and ES which clearly distinguishes the former from the latter:

- In EP, mutation is the only adaptation operation applied to the individuals in the population. *No* recombination operations are carried out.

- Instead of using a deterministic selection strategy as in ES, where γ descendent optimizers are created from μ parent optimizers, and the members of the new population selected from the resulting combination according to a ranking of their respective fitness values, EP uses a stochastic selection strategy as follows: for each optimizer in the $(\mu + \gamma)$ parent/descendent combination, we randomly select Q other optimizers in the same combination and compare their fitness values with the current optimizer. The current optimizer is included in the population in the next generation if its fitness value is greater than those of the Q "opponents" selected. In addition, the number of descendants γ is usually set equal to the number of parents μ. In other words, this *Q-tournament selection strategy* can be regarded as a probabilistic version of the $(\mu + \mu)$ selection strategy in ES.

The tournament selection strategy has the advantage that, compared with the deterministic selection strategy, even optimizers which are ranked in the lower half of the $(\mu+\mu)$ parent and descendent combination have a positive probability of being included into the population in the next generation. This is especially useful for nonstationary fitness functions when certain optimizers in the population which at first seem non-promising turn out to be highly relevant due to the constant evolving nature of the fitness landscape. These optimizers may probably be already excluded in the initial stages of optimization if a deterministic strategy is adopted, but in the case of a tournament selection strategy, it is still possible for these optimizers to be included if there are indications that they are slightly distinguished from the truly non-promising optimizers through the result of the tournament competition.

This is also the reason for our choice of EP for our adaptive regularization problem: in our algorithm, we have generated a population of *regularization strategies*, which are vectors of regularization and segmentation parameters, as our potential optimizers. The optimal regularization strategy is selected from the population at each generation according to criteria to be described in a later section, and this is used to restore the image for a single iteration. This partially restored image is then used as the basis for the evolution of regularization strategies in the next generation. In other words, the fitness landscape in the next generation depends on the particular optimizer selected in the previous generation, thus constituting a non-stationary optimization problem. The stochastic tournament selection strategy is therefore useful in retaining those regularization strategies which initially seem non-promising but are actually highly relevant in later restoration stages.

7.3 The ETC-pdf Image Model

In this Chapter, we address the adaptive regularization problem by proposing a novel image model, the adoption of which in turn necessitates the use of powerful optimization algorithms such as those typical in the field of evolutionary computation. This model is

observed to be capable of succinctly characterizing common properties of a large class of images, and thus we will regard any restored image which conforms to this image model as suitably regularized. In other words, this model can be regarded as a possible objective characterization of our usual notion of subjective quality. The model, which is specified as the probability distribution of the ETC measure, is approximated as a histogram, and the regularization parameters in the various image regions are chosen in such a way that the corresponding ETC-histogram of the resulting restored image matches the model pdf closely, i.e., minimizing the difference between the two distributions. It is obvious that the resulting error function, which involves differences of discrete distributions, is highly irregular and non-differentiable, and thus necessitates the use of powerful optimization algorithms. In view of this, we have chosen evolutionary programming as the optimization algorithm to search for the minimizer of this cost function.

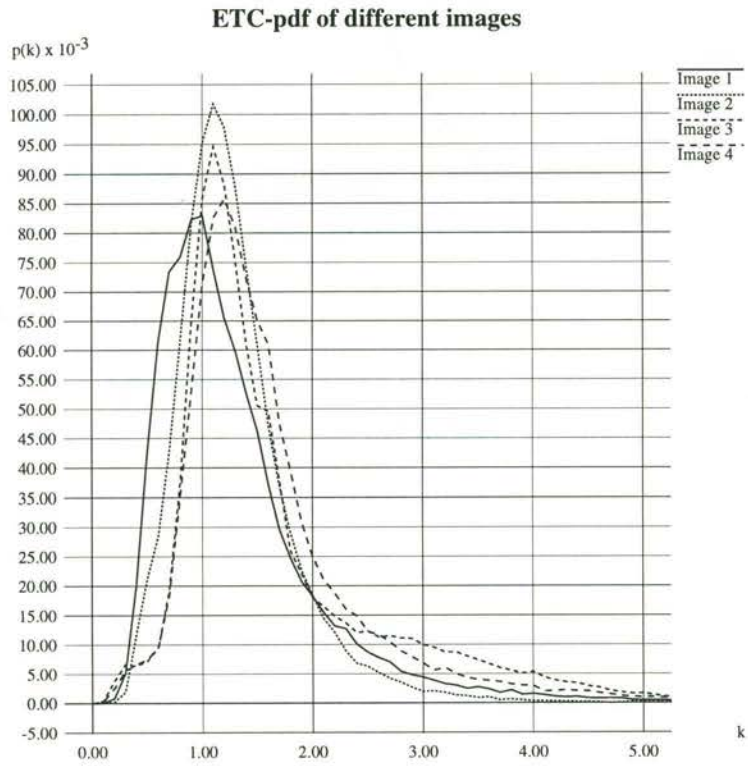


Figure 7.1: The ETC-pdf of different images

Denoting the probability density function of the ETC measure κ within a typical image as $p_\kappa(\kappa)$, we have plotted the ETC-histograms, which are approximations of the

ETC-pdf, for several images in Figure 7.1. It is noticed that the histograms peak around $\kappa = 1$, indicating the predominance of smooth regions. As κ increases, values of the various histograms gradually decrease, with $p_\kappa(\kappa) \approx 0$ for $\kappa \approx K$, which is the size of the averaging window used, ($K = 5$ in the current example). This indicates the smaller proportion of edges and textured regions. More importantly, it is seen that, although there are slight deviations between the various ETC-histograms, they in general assume the typical form as shown in Figure 7.1. This is in contrast with the traditional gray-level histogram which is usually used to characterize the gray level distribution in an image, and which can vary widely from image to image. This is illustrated in Figures 7.2(a) to (d), where the gray level histograms for the same images are shown. As a result, we can consider the ETC-histogram as a form of *signature* for a large class of non-degraded images.

On the other hand, it is observed that the corresponding density functions for degraded images are usually very different from the standard density function. Figure 7.3 illustrates this point by comparing the corresponding ETC-pdf of one of the images in Figure 7.1 and its blurred version. In the figure, the solid curve is the original ETC-pdf and the dotted curve is the ETC-pdf of the blurred image. It is seen that the rate of decrease is greater for the blurred image, indicating the higher degree of correlation among its pixels. Therefore, one possible regularization strategy to allow the restored image to more closely resemble the original image is to assign the parameters in such a way as to minimize the discrepancies between the ETC-histogram of the restored image and that of the original.

Due to the similar shapes of the ETC-pdf in Figure 7.1, we can model the ETC-pdf using a combination of piecewise Gaussian and exponential functions. During restoration, we can adaptively assign the regularization parameters in such a way that the corresponding ETC-pdf in the restored image conforms closely to the model density function. In this work, we have adopted the following model $p_\kappa^M(\kappa)$ for the typical ETC-pdf, which

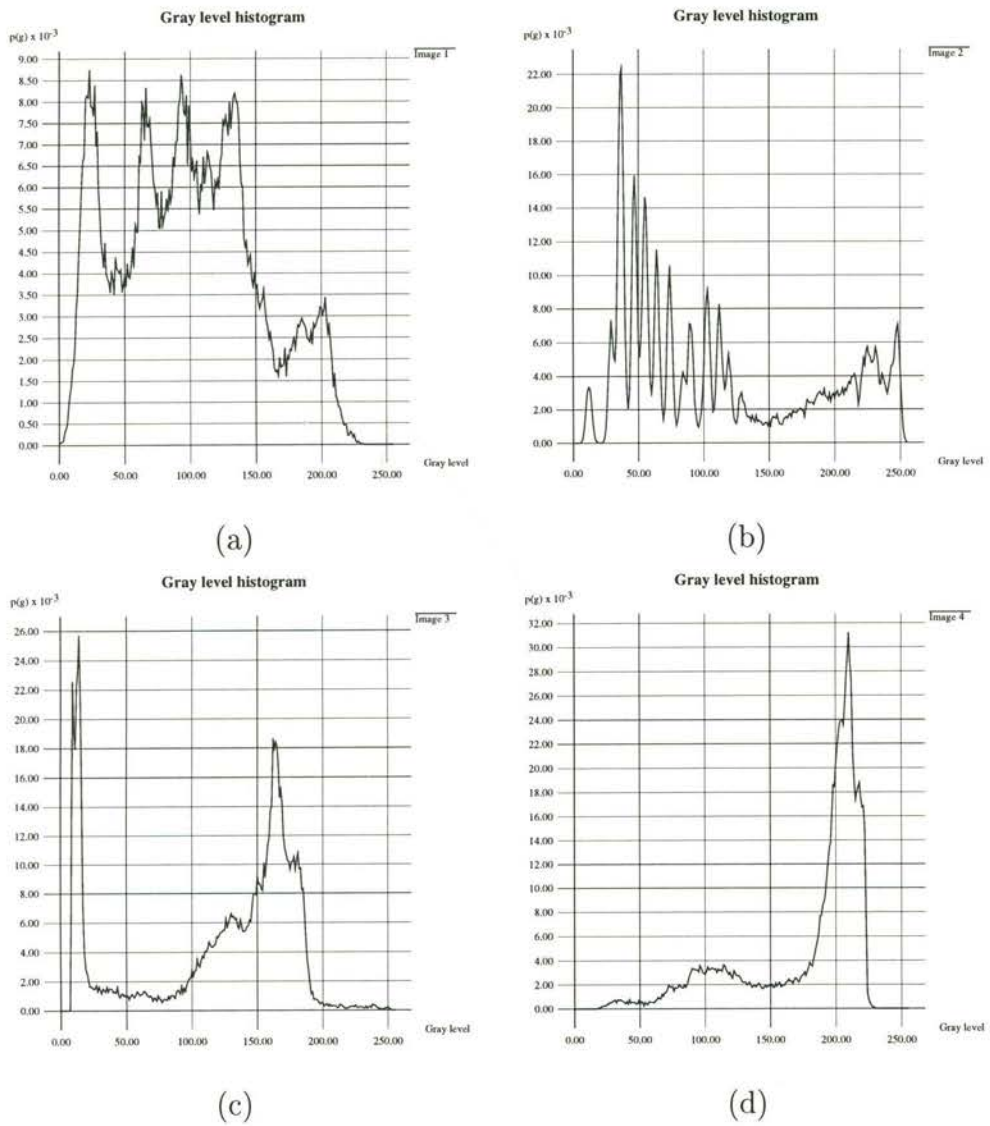


Figure 7.2: Gray level histograms of different images (a) Image 1 (b) Image 2 (c) Image 3 (d) Image 4

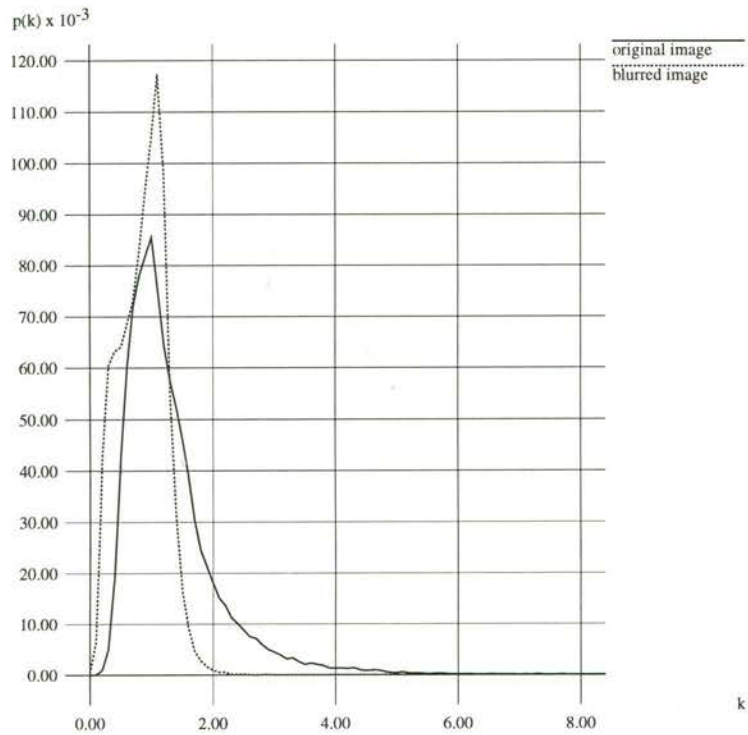


Figure 7.3: The ETC-pdf for a typical image and its blurred version. Solid curve is the ETC-pdf of the original image. Dotted curve is the ETC-pdf of the blurred image. Size of averaging window is 5×5.

can best characterize the density function of a large class of images.

$$p_\kappa^M(\kappa) = \begin{cases} ce^{-(\kappa-1)^2} & 0 \le \kappa \le 1 \\ ca_1^{\varepsilon(\kappa-1)} & 1 < \kappa \le \kappa_3 \\ ca_1^{\varepsilon(\kappa_3-1)} a_2^{\varepsilon(\kappa-\kappa_3)} & \kappa_3 < \kappa \le K \end{cases} \tag{7.6}$$

In this equation, we use a Gaussian function segment to model the density function when $\kappa \in [0, 1]$, and use two exponential function segments to model the tail distribution in the interval $[1, K]$, with the corresponding parameters a_1 and a_2 satisfying $a_1 < a_2$. The constant κ_3 in the equation is that value of κ which corresponds to an almost equipartition of the K^2 variables into three components as described in Chapter 5. It has been shown for the case of 5×5 averaging window that $\kappa_3 \approx 1.8$. We could have modelled the density function in the interval $[1, K]$ using a single exponential segment with parameter a, and in turn estimate this parameter from a histogram of κ using typical real-world images. Instead, we have used two separate exponential functions, with a_1 and a_2 chosen such that $a_1 < a_2$. The reason for doing this and choosing the particular transition point κ_3 is that: for $\kappa \in [1, \kappa_3]$, the neighborhood surrounding the current pixel consists of less than three gross components, which usually corresponds to a mixture of noises and smooth regions, and is particularly undesirable given the enhanced visibility of noises against the smooth background. One must therefore limit the probability of occurrence of such values of κ, which explains the adoption of $a_1 < a_2$ in the probability density model allowing a smaller probability of occurrences for $\kappa \in [1, \kappa_3]$. One may argue that this may adversely affect the probability of edge occurrence, which consists of two gross components with $\kappa = \kappa_2 \in [1, \kappa_3]$ as well, but edges usually occupy a small area in a typical image, which translates to a very small occurrence probability, so it is much more probable that those locations with κ in that interval correspond to a mixture of noises and smooth backgrounds, and the main effect of probability reduction in this interval is the elimination of this type of artifacts. The variable ε controls the rates of decay for the two exponentials, and the constant c in the equation is a normalization factor such that

$$\int_0^K p_\kappa^M(\kappa)\, d\kappa = 1 \tag{7.7}$$

7.4 Adaptive Regularization Using Evolutionary Programming

As mentioned previously, the value of the regularization parameter in image restoration applications is usually determined by trial and error. Therefore, the purpose of establishing the model in Section 7.3 is to provide an objective way by which we can assign the parameters adaptively to result in the best subjective quality. Formally, we replace the constrained least square cost function for image restoration in Chapter 3 with the following cost function:

$$E = \frac{1}{2}\|\mathbf{y} - \mathbf{H}\hat{\mathbf{x}}\|^2 + \frac{1}{2}\|\mathbf{D}\hat{\mathbf{x}}\|^2_{\mathbf{L}} \tag{7.8}$$

where, instead of a single parameter λ, we employ a diagonal *weighting matrix* $\mathbf{L}$ defined as follows:

$$\mathbf{L} = diag[\lambda(\sigma_1), \ldots\ldots, \lambda(\sigma_{N_I})] \tag{7.9}$$

where $\sigma_i, i = 1, \ldots, N_I$ is the local standard deviation of the i-th pixel in the image. This is similar to the case of the spatially adaptive iterative restoration algorithm by Kang and Katsaggelos [53, 56] , described in the experimental section in Chapter 6, where we employ the following form for $\mathbf{L}$

$$\mathbf{L} = \lambda\ diag[1 - \frac{\sigma_1^2}{\sigma^2_{\max}}, \ldots\ldots, 1 - \frac{\sigma^2_{N_I}}{\sigma^2_{\max}}] \tag{7.10}$$

and $\sigma_{\max}$ denotes the maximum value among all the local standard deviations. (The local variances σ_i should be distinguished from the mutation strategy parameter σ_i^m). In that algorithm, an additional weighting matrix $\mathbf{A}(\hat{\mathbf{x}})$ is also adopted for the first data term, while for the present case no such weightings are employed. In addition, whereas the diagonal entries of $\mathbf{L}$ for the KK-SAI algorithm is restricted to the form in equation (7.10) , the corresponding entries $\lambda(\sigma_i)$ for the current algorithm can be any arbitrary function of the local standard deviation σ_i

Denote the ETC-pdf for the restored image $\hat{\mathbf{x}}$ as $p_\kappa(\kappa)$, our objective is to select the particular forms for $\lambda(\sigma_i)$ in the weighting matrix $\mathbf{L}$ in such a way to minimize the

following *weighted probability density error measure* of the ETC measure κ (ETC-pdf error measure).

$$E_{pdf}^{\kappa} = \int_0^K w(\kappa)(p_{\kappa}^M(\kappa) - p_{\kappa}(\kappa))^2 \, d\kappa \tag{7.11}$$

where the weighting coefficients are defined as follows

$$w(\kappa) \equiv \frac{1}{\max(p_{\kappa}^M(\kappa), p_{\kappa}(\kappa))^2} \tag{7.12}$$

to compensate for the generally smaller contribution of the tail region to the total probability. In practice, we replace the integral operation by a finite summation over a suitable discretization of $[0, K]$, and the density function $p_{\kappa}(\kappa)$ is approximated using the histogram of κ in the partially restored image.

$$E_{pdf}^{\kappa} = \sum_{r=1}^{d} w(r\triangle)(p_{\kappa}^M(r\triangle) - \hat{p}_{\kappa}(r\triangle))^2 \tag{7.13}$$

where $\triangle$ is the width of the discretization interval, $\hat{p}_{\kappa}(r\triangle)$ is the estimated ETC-pdf of κ in terms of its histogram , and d is the number of discretization intervals. Since the histogram records the relative occurrence frequencies of the various values of κ based on the previous discretization, it involves the counting of discrete quantities. As a result, the overall error function (7.13) is obviously non-differentiable with respect to the regularization parameters, and the evolutionary programming approach provides a viable option to minimize this error function.

Evolutionary programming is a population-based optimization algorithm in which the individual optimizers in the population compete against each other with respect to the minimization of a cost function, or equivalently, the maximization of a fitness function [8]. In the current case, we have already specified the fitness function, which is equation (7.13). In addition, we have to specify the form of the individual optimizers in the population as well. For the adaptive regularization problem, we consider the following *regularization profile* $\lambda(\sigma_i)$ defined on the local standard deviation σ_i around the i-th pixel

$$\lambda(\sigma_i) = \frac{\lambda_{\max} - \lambda_{\min}}{1 + e^{\beta(\sigma_i - \alpha)}} + \lambda_{\min} \tag{7.14}$$

Equation (7.14) defines a decreasing sigmoidal function on the local standard deviation range of the image, which is consistent with our previous view that large λ is required at

low variance pixels to suppress noise and small λ is required at high variance pixels to enhance the features there. There are four parameters in equation (7.14) which determines the overall λ assignment strategy: $\lambda_{\min}$ and $\lambda_{\max}$ represent the minimum and maximum parameter values used respectively, α represents the offset of the sigmoidal transition from the origin, thus implicitly defining the standard deviation threshold which separates the small variance from the large variance region. β controls the steepness of the sigmoidal transition. The various parameters of a typical regularization profile are illustrated in Figure 7.4.

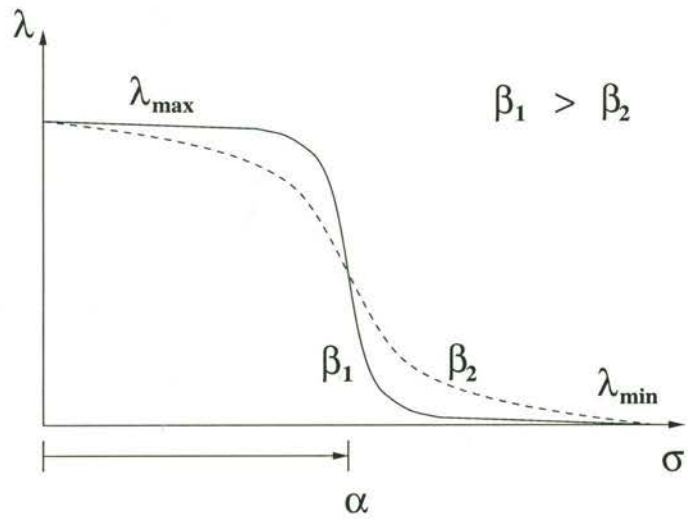


Figure 7.4: Illustrating the various parameters of a typical regularization profile $\mathcal{S}_p$

Concatenating these four parameters together with the *mutation strategy parameters* $\sigma^m_{\lambda_{\min}}, \sigma^m_{\lambda_{\max}}, \sigma^m_\alpha, \sigma^m_\beta$ (not to be confused with the local standard deviation σ_i of an image) into an 8-tuple, we define the following *regularization strategy* $\mathcal{S}_p$ as the p-th potential optimizer in the population.

$$\mathcal{S}_p \equiv (\lambda_{\min,p}, \lambda_{\max,p}, \alpha_p, \beta_p, \sigma^m_{\lambda_{\min},p}, \sigma^m_{\lambda_{\max},p}, \sigma^m_{\alpha,p}, \sigma^m_{\beta,p}) \tag{7.15}$$

Employing the usual operations of evolutionary computational algorithms, which in the case of evolutionary programming is restricted to the mutation operator [29], we generate a population P consisting of μ instances of $\mathcal{S}_p$ in the first generation, and we apply mutation to each of these μ parents to generate μ descendants in each subsequent generation

according to the following equations

$$\lambda'_{\min,p} = \lambda_{\min,p} + N(0, \sigma^m_{\lambda_{\min,p}}) \tag{7.16}$$

$$\lambda'_{\max,p} = \lambda_{\max,p} + N(0, \sigma^m_{\lambda_{\max,p}}) \tag{7.17}$$

$$\alpha'_p = \alpha_p + N(0, \sigma^m_{\alpha,p}) \tag{7.18}$$

$$\beta'_p = \beta_p + N(0, \sigma^m_{\beta,p}) \tag{7.19}$$

where $N(0, \sigma)$ denotes a Gaussian random variable with zero mean and standard deviation σ. The variables $\lambda'_{\min,p}$, $\lambda'_{\max,p}$, α'_p and β'_p are the components of the descendant strategy $\mathcal{S}'_p$. The mutation strategy parameters are updated according to the following *log-normal adaptation rule* .

$$\sigma'^m_{\lambda_{\min},p} = \sigma^m_{\lambda_{\min},p} \exp(N(0, \tau_1) + N_j(0, \tau_2)) \tag{7.20}$$

$$\sigma'^m_{\lambda_{\max},p} = \sigma^m_{\lambda_{\max},p} \exp(N(0, \tau_1) + N_j(0, \tau_2)) \tag{7.21}$$

$$\sigma'^m_{\alpha,p} = \sigma^m_{\alpha,p} \exp(N(0, \tau_1) + N_j(0, \tau_2)) \tag{7.22}$$

$$\sigma'^m_{\beta,p} = \sigma^m_{\beta,p} \exp(N(0, \tau_1) + N_j(0, \tau_2)) \tag{7.23}$$

where $N(0, \tau_1)$ is held fixed across all mutation strategy parameters, while $N_j(0, \tau_2)$ is generated anew for each parameter. The values for τ_1 and τ_2 are $(\sqrt{2N})^{-1}$ and $(\sqrt{2\sqrt{N}})^{-1}$ respectively, as suggested in [7], where $2N$ is the dimension of the regularization strategy $\mathcal{S}_p$.

For each regularization strategy $\mathcal{S}_p$ in the population, we can in principle use it to restore a blurred image, and then build up the histogram $\hat{p}_\kappa(\kappa)$ by evaluating κ at each pixel and counting their relative occurrence frequencies. Finally, we can evaluate the ETC-pdf error $E^\kappa_{pdf}(\mathcal{S}_p)$ as a function of the strategy $\mathcal{S}_p$ by comparing the difference of the normalized histogram and the model ETC-pdf, and use this value as our basis for competition and selection. Adhering to the canonical operations of EP, we employ tournament competition as our chief selection mechanism. For tournament competition, we generate a subset $\mathcal{T}(\mathcal{S}_p) \subset P \setminus \{\mathcal{S}_p\}$ for each strategy $\mathcal{S}_p$ with $|\mathcal{T}| = Q$ by randomly sampling the population Q times. We can then form the set $\mathcal{W}(\mathcal{S}_p)$ as below

$$\mathcal{W}(\mathcal{S}_p) = \{\mathcal{S}_q \in \mathcal{T} : E^\kappa_{pdf}(\mathcal{S}_q) > E^\kappa_{pdf}(\mathcal{S}_p)\} \tag{7.24}$$

which contains all those strategies $\mathcal{S}_q$ in $\mathcal{T}$ with their corresponding ETC-pdf error greater than that of $\mathcal{S}_p$. We then define the *win count* $w_c(\mathcal{S}_p)$ as the cardinality of $\mathcal{W}(\mathcal{S}_p)$

$$w_c(\mathcal{S}_p) = |\mathcal{W}(\mathcal{S}_p)| \tag{7.25}$$

Denoting the population at the current generation as $P(t)$, we order the regularization strategies $\mathcal{S}_p$ in $P(t)$ according to decreasing values of their associated win counts $w_c(\mathcal{S}_p)$, and choose the first μ individuals in the list to be incorporated into the next generation $P(t+1)$.

7.4.1 Competition Under Approximate Fitness Criterion

As we have mentioned, we can in principle perform a restoration for each of the 2μ regularization strategies in $P(t)$ to evaluate its fitness , but each restoration is costly in terms of the amount of computations, since it has to be implemented iteratively, and is especially the case when the number of individuals in the population is large. We therefore resort to an approximate competition and selection process where each individual strategy in the population is used to restore only a part of the image. Specifically, we associate each strategy $\mathcal{S}_p, p = 1, \ldots, 2\mu$ in the population $P(t)$ with a subset $\mathcal{R}_p \subset \mathcal{X}$, where $\mathcal{X}$ denotes the set of points in an $N_y \times N_x$ image lattice

$$\mathcal{X} = \{(i_1, i_2) : 1 \leq i_1 \leq N_y, 1 \leq i_2 \leq N_x\} \tag{7.26}$$

and where the regions $\mathcal{R}_p$ form a partition of $\mathcal{X}$

$$\mathcal{R}_{p_1} \cap \mathcal{R}_{p_2} = \emptyset \quad p_1 \neq p_2 \tag{7.27}$$

$$\bigcup_{p=1}^{2\mu} \mathcal{R}_p = \mathcal{X} \tag{7.28}$$

In this way, we can define the quantity $\hat{E}^{\kappa}_{pdf}(\mathcal{S}_p, \mathcal{R}_p)$, which is both a function of $\mathcal{S}_p$ and $\mathcal{R}_p$, as the evaluation of $E^{\kappa}_{pdf}(\mathcal{S}_p)$ restricted to the subset $\mathcal{R}_p$ only. This serves as an estimate of the exact $E^{\kappa}_{pdf}(\mathcal{S}_p)$, which is evaluated over the entire image lattice $\mathcal{X}$ and is therefore a function of $\mathcal{S}_p$ only. In order to approximate the original error function closely, we must choose the subsets $\mathcal{R}_p$ in such a way that each of them captures the essential

features of the distribution of the ETC measure in the complete image. This implies that each of these subsets should be composed of disjoint regions scattered almost uniformly throughout the lattice $\mathcal{X}$ and that both the ways of aggregating these regions into a single $\mathcal{R}_p$ and the ways of assigning these subsets to each individual in the population should be randomized in each generation. In view of these, we partition the image lattice $\mathcal{X}$ into non-overlapping square blocks B_s of size $n \times n$

$$\mathcal{B} = \{B_s : s \in \mathcal{I}\} \tag{7.29}$$

where

$$B_{s_1} \cap B_{s_2} = \emptyset \quad s_1 \neq s_2 \tag{7.30}$$

$$\bigcup_{s \in \mathcal{I}} B_s = \mathcal{X} \tag{7.31}$$

In this equation, $\mathcal{I}$ is the index set $\{1, \ldots, S\}$ and S is the number of $n \times n$ square blocks in the image lattice. Assuming $2\mu < S$ and furthermore that $u \equiv S/2\mu$ is an integer, i.e., 2μ is a factor of S, we can construct the following *random* partition $\mathcal{P}_{\mathcal{I}} = \{\mathcal{I}_p : p = 1, \ldots, 2\mu\}$ of $\mathcal{I}$ where

$$\mathcal{I}_{p_1} \cap \mathcal{I}_{p_2} = \emptyset \quad p_1 \neq p_2 \tag{7.32}$$

$$\bigcup_{p=1}^{2\mu} \mathcal{I}_p = \mathcal{I} \tag{7.33}$$

and $|\mathcal{I}_p| = u$ as follows:

- Randomly sample the index set $\mathcal{I}$ u times without replacement to form $\mathcal{I}_1$.
- For $p = 2, \ldots, 2\mu$, randomly sample the set $\mathcal{I} \setminus \bigcup_{q=1}^{p-1} \mathcal{I}_q$ u times to form $\mathcal{I}_p$

Finally, we define the subsets $\mathcal{R}_p$ corresponding to each individual strategy $\mathcal{S}_p$ as follows.

$$\mathcal{R}_p = \bigcup_{s \in \mathcal{I}_p} B_s \tag{7.34}$$

We can then evaluate $\hat{E}^{\kappa}_{pdf}(\mathcal{S}_p, \mathcal{R}_p)$ for each strategy $\mathcal{S}_p$ with its corresponding region $\mathcal{R}_p$ and use it in place of the exact error $E^{\kappa}_{pdf}(\mathcal{S}_p)$ for the purpose of carrying out the tournament competition and ranking operation in Section 7.4. The process is illustrated

in Figure 7.5. In this case, for the purpose of illustration, we have used a population with only four individuals. The image is divided into 12 sub-blocks and we have assigned 3 sub-blocks to each individual in the population. This assignment is valid only for one update iteration only, and a randomly generated alternative assignment is adopted for the next iteration to ensure adequate representation of the entire image through these assigned blocks.

1	**4**	**3**	**2**
3	**4**	**1**	**3**
4	**1**	**2**	**2**

Figure 7.5: Illustrating the assignment of regions $\mathcal{R}_p$ to individual regularization strategy $\mathcal{S}_p$ for a 4-member population.

7.4.2 Choice of optimal regularization strategy

After the tournament competition and ranking operations, we can in principle choose the optimal strategy $\mathcal{S}^*$ from the population $P(t)$ and use it to restore the image for a single iteration. We can define optimality here in several ways. The obvious definition is to choose that strategy $\mathcal{S}^*$ with the minimum value of $\hat{E}^{\kappa}_{pdf}$

$$\mathcal{S}^* = \arg\min_{p} \hat{E}^{\kappa}_{pdf}(\mathcal{S}_p, \mathcal{R}_p) \tag{7.35}$$

Alternatively, we can form a subset of those elite strategies with their win index $w_c(\mathcal{S}_p)$ equal to the maximum possible value, i.e., the tournament size Q:

$$\mathcal{E}_Q = \{\mathcal{S}_p : w_c(\mathcal{S}_p) = Q\} \subset P(t) \tag{7.36}$$

and then choose our strategy $\mathcal{S}^*$ by uniformly sampling from this subset.

The above selection schemes are suitable for the competition and selection stage of conventional evolutionary computational algorithms where the fitness value of each indi-

vidual directly reflects its inherent optimality. In the current approximate competition scheme, however, we have replaced the exact fitness value E^{κ}_{pdf} with the estimate $\hat{E}^{\kappa}_{pdf}$, which depends on both the inherent fitness of the strategy $\mathcal{S}_p$ and its particular assigned region $\mathcal{R}_p$. Therefore, there may exist cases where a non-optimal strategy will acquire a high fitness score due to a fortuitous combination of image blocks B_s in forming its assigned region. To prevent this, a more sophisticated selection scheme involving the elite individuals in the population is required.

Recalling that the estimated error $\hat{E}^{\kappa}_{pdf}$ is both a function of the inherent fitness of the strategy $\mathcal{S}_p$ and its assigned region $\mathcal{R}_p$, and a fortuitous combination of image blocks B_s may result in a low estimated error even though the inherent fitness of $\mathcal{S}_p$ is not very high. But the very word "fortuitous" implies the rarity of this event, and since the assigned region for each regularization strategy changes in each generation, we should expect that for a strategy with low inherent fitness, it will quickly encounter a combination of image blocks $\mathcal{B}_s$ leading to a very large estimated error and become displaced from the population. On the other hand, for a strategy with high inherent fitness, we should expect that the corresponding estimated error will be low for a variety of image block combinations in each generation, and it will most probably survive into the next generation. In view of this, a proper way to estimate an optimal regularization strategy from the current population should involve the *survival time* t_p of each individual strategy $\mathcal{S}_p$ which is defined as follows:

$$t_p \equiv t - t^i_p \tag{7.37}$$

such that

$$\mathcal{S}_p \in \bigcap_{t'=t^i_p}^{t} P(t') \tag{7.38}$$

In other words, t^i_p is the generation where the strategy $\mathcal{S}_p$ first appears in the population, and t is the current generation. The survival time t_p is defined as the difference between these two indices. In general, it is reasonable to assume that a regularization strategy with a long survival time t_p is more likely to possess high inherent fitness, but whether a strategy with a short survival time possesses high inherent fitness is yet to be confirmed in

later generations. As a result, it is more reasonable to choose the optimal regularization strategy based on those in the population with long survival times. Rearranging the corresponding survival time t_p of each individual in ascending order,

$$t_{(1)}, \ldots, t_{(p)}, \ldots, t_{(2\mu)} \tag{7.39}$$

such that

$$t_{(1)} \leq \ldots \leq t_{(p)} \leq \ldots \leq t_{(2\mu)} \tag{7.40}$$

and $t_{(p)}$ denotes the p-th order statistic of the survival time sequence, we expect those values $t_{(p)}$ with large index p will most likely correspond to a strategy $\mathcal{S}_{(p)}$ with high inherent fitness. Choosing $p_0 > 1$ and regarding each strategy as a vector in $\mathbf{R}^8$, we define the following *combined regularization strategy*

$$\mathcal{S}^*_{p_0} = \frac{1}{2\mu - p_0 + 1} \sum_{p=p_0}^{2\mu} \mathcal{S}_{(p)} \tag{7.41}$$

To ensure the inclusion of only those individuals with high inherent fitness in the above averaging operation, the value p_0 is usually chosen such that $p_0 \gg 1$.

In addition, if we possess *a priori* knowledge regarding inherent properties of desirable regularization strategies characterized by the *constraint set* $\mathcal{C}_\mathcal{S}$, we can modify the previous averaging procedure to include this knowledge as follows

$$\mathcal{S}^*_{p_0} = \frac{\sum_{p=p_0}^{2\mu} I_{\{\mathcal{S}_{(p)} \in \mathcal{C}_\mathcal{S}\}} \mathcal{S}_{(p)}}{\sum_{p=p_0}^{2\mu} I_{\{\mathcal{S}_{(p)} \in \mathcal{C}_\mathcal{S}\}}} \tag{7.42}$$

where $I_{\{\mathcal{S}_{(p)} \in \mathcal{C}_\mathcal{S}\}}$ is the indicator function of $\mathcal{C}_\mathcal{S}$.

The optimal strategy $\mathcal{S}^*_{p_0}$ is constructed based on the estimated ETC-pdf error $\hat{E}^\kappa_{pdf}$, and we have to evaluate its performance based on the true error measure E^κ_{pdf} eventually, which requires using $\mathcal{S}^*_{p_0}$ to restore the whole image for one iteration and then collecting the statistics of the measure κ on the resulting image to form an estimation of $p_\kappa(\kappa)$. In the current algorithm, this has to be performed once only, again highlighting the advantage of using $\hat{E}^\kappa_{pdf}$, which requires performing restoration over an image subset only for each individual strategy in the population, instead of using E^κ_{pdf} in the competition process, which requires performing restoration for an entire image . In this way, the time required for fitness evaluation is independent of the number of individuals in the population.

Denoting the current optimal strategy as $\mathcal{S}^*(t)$ and the last strategy used to update the image as $\mathcal{S}^*(t-1)$, we should decide between using $\mathcal{S}^*_{p_0}$ or $\mathcal{S}^*(t-1)$ as $\mathcal{S}^*(t)$, or we should simply left the image unrestored for the present iteration. Denoting the *null strategy* as $\mathcal{S}_\phi$, which amount to leaving the present image unrestored, the basis of this decision should be the true ETC-pdf error value $E^\kappa_{pdf}(\mathcal{S})$ where $\mathcal{S} = \mathcal{S}^*_{p_0}, \mathcal{S}^*(t-1)$ or $\mathcal{S}_\phi$. Using $\hat{\mathbf{x}}_r(\mathcal{S})$ to indicate the resulting restored image through the action of $\mathcal{S}$, $\hat{\mathbf{x}}(t)$ as the updated image, and $\hat{\mathbf{x}}(t-1)$ as the pre-updated image, we adopt the following decision rule for choosing $\mathcal{S}^*(t)$

- If $E^\kappa_{pdf}(\mathcal{S}^*_{p_0}) = \min\{E^\kappa_{pdf}(\mathcal{S}^*_{p_0}), E^\kappa_{pdf}(\mathcal{S}^*(t-1)), E^\kappa_{pdf}(\mathcal{S}_\phi)\}$
  1. $\mathcal{S}^*(t) = \mathcal{S}^*_{p_0}$
  2. $\hat{\mathbf{x}}(t) = \hat{\mathbf{x}}_r(\mathcal{S}^*_{p_0})$
- If $E^\kappa_{pdf}(\mathcal{S}^*(t-1)) = \min\{(E^\kappa_{pdf}(\mathcal{S}^*_{p_0}), E^\kappa_{pdf}(\mathcal{S}^*(t-1)), E^\kappa_{pdf}(\mathcal{S}_\phi)\}$
  1. $\mathcal{S}^*(t) = \mathcal{S}^*(t-1)$
  2. $\hat{\mathbf{x}}(t) = \hat{\mathbf{x}}_\mathbf{r}(\mathcal{S}^*(t-1))$
- If $E^\kappa_{pdf}(\mathcal{S}_\phi) = \min\{(E^\kappa_{pdf}(\mathcal{S}^*_{p_0}), E^\kappa_{pdf}(\mathcal{S}^*(t-1)), E^\kappa_{pdf}(\mathcal{S}_\phi)\}$
  1. $\mathcal{S}^*(t) = \mathcal{S}_\phi$
  2. $\hat{\mathbf{x}}(t) = \mathbf{x}_r(\mathcal{S}_\phi) = \mathbf{x}(t-1)$

The convergence of the general evolutionary programming algorithm was demonstrated in [29]. However, the current modified form of EP depends more on the monotonically decreasing value of $E^\kappa_{pdf}(\mathcal{S})$ in the above implementation. Together with the fact that $E^\kappa_{pdf}(\mathcal{S})$ is bounded below by zero, the current algorithm is guaranteed to converge.

7.5 Incorporation of ETC Fuzzy Criterion

To further improve the restoration result, we can incorporate the ETC fuzzy membership values for selective regularization of high variance pixels as either edges or textures. The

chief motivation of including this additional step is due to our observation that the parameter $\lambda_{\min}$ corresponding to the final evolved regularization strategy is usually biased in favor of textures. Using the sigmoid regularization function $\lambda(\sigma_i)$ (or adopting a 2-D index, $\lambda(\sigma_{i_1,i_2})$), for adaptive parameter assignment, those pixels with high standard deviation $\sigma_{i_1,i_2} > \alpha$ (which may either be textures or edges) are essentially assigned with a parameter value $\lambda \approx \lambda_{\min}$. While this value is near optimal for the textured regions, it is usually too small for regularizing the edges which are not as effective in masking noises, thus resulting in a noisy appearance around the edges. This is analogous to the case in HMBNN regularization where we employed texture-oriented prediction scheme to estimate the desired output for the restoration network. While retaining the value of $\lambda \approx \lambda_{\min}$ for the texture pixels, we would like to adopt $\lambda > \lambda_{\min}$ for the edge pixels.

This can be accomplished in the present case by simply defining two parameter values, $\lambda^{edge}_{i_1,i_2}$ for edges and $\lambda^{tex}_{i_1,i_2}$ for textures, at each pixel (i_1, i_2), and requiring that $\lambda^{edge}_{i_1,i_2} > \lambda^{tex}_{i_1,i_2}$. Unlike the fuzzy HMBNN algorithm where we separately estimate the two parameter values, here we adopt a simpler approach where we just require that $\lambda^{edge}_{i_1,i_2}$ be proportional to $\lambda^{tex}_{i_1,i_2}$. This is a reasonable assumption since a greater $\lambda^{tex}_{i_1,i_2}$ discovered by the evolutionary algorithm, which signifies a more severely degraded image, usually implies the requirement of a greater $\lambda^{edge}_{i_1,i_2}$ for the edges as well. More formally, we define

$$\lambda^{tex}_{i_1,i_2} = \lambda(\sigma_{i_1,i_2}) \tag{7.43}$$

$$\lambda^{edge}_{i_1,i_2} = \zeta\lambda^{tex}_{i_1,i_2} \tag{7.44}$$

where σ_{i_1,i_2} is the local standard deviation at (i_1, i_2), and $\zeta > 1$ is a proportional factor.

For each high variance pixels, we would then evaluate the two ETC fuzzy coefficient values, $\widetilde{\mu}_E(\kappa_{i_1,i_2})$ and $\widetilde{\mu}_T(\kappa_{i_1,i_2})$ according to the local ETC measure κ_{i_1,i_2}. The final lambda value λ_{i_1,i_2} is then obtained as a convex combination of $\lambda^{edge}_{i_1,i_2}$ and $\lambda^{tex}_{i_1,i_2}$.

$$\lambda_{i_1,i_2} = \widetilde{\mu}_E(\kappa_{i_1,i_2})\lambda^{edge}_{i_1,i_2} + \widetilde{\mu}_T(\kappa_{i_1,i_2})\lambda^{tex}_{i_1,i_2} \tag{7.45}$$

This is in contrast with the previous HMBNN case where, instead of combining the two λ values, we combine the two *required gray level update values* $\triangle\widetilde{v}^{edge}_{i_1,i_2}$ and $\triangle\widetilde{v}^{tex}_{i_1,i_2}$. While

this is more natural in the previous neural network setting where we are considering the required updates as hidden neuron outputs, the ETC fuzzy coefficient values as output weights, and the final update $\triangle\widetilde{v}_{i_1,i_2}$ as being the output of a linear combiner output neuron, it is more appropriate to apply the fuzzy combination on the parameter λ in this case, which is considered as the emergent output of the evolutionary process.

7.6 Experimental results

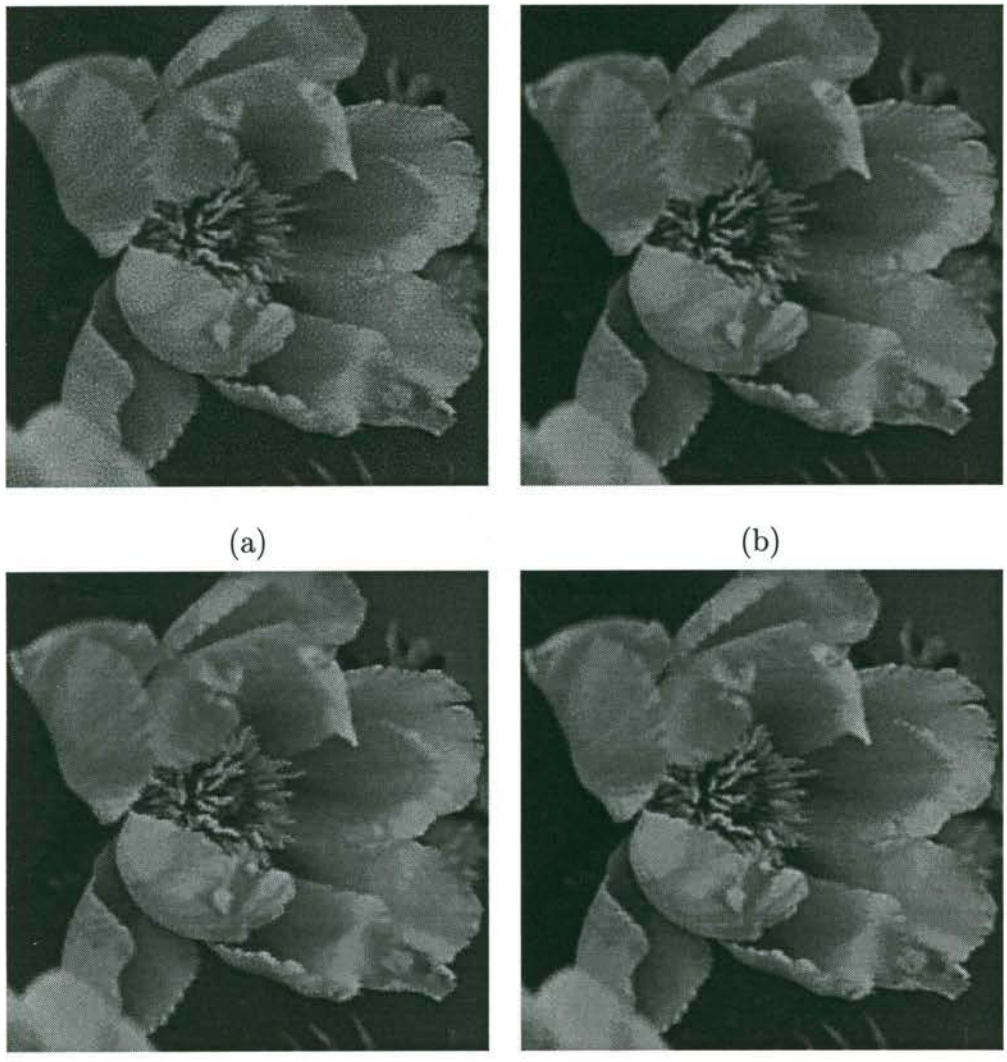
As in previous experiments on adaptive regularization, we applied the current evolutionary algorithm to the three images flower, Lena and eagle (the original images are shown in Figures 3.3 in Chapter 3). Progressive degrees of degradation including 5×5 Gaussian blur ($\sigma_g = 1$) at 30dB BSNR, 5 × 5 uniform blur respectively at 30dB BSNR and 20dB BSNR are applied to the images to test the robustness of the current algorithm.

The parameters of the ETC-pdf image model were chosen as follows: $a_1 = 0.83, a_2 = 0.85$ and $\kappa_3 = 1.8$ using a 5×5 averaging window. The value $\triangle = 0.1$ is used as the width of the histogram discretization interval. The parameter ε in the model is chosen to be $\frac{1}{\triangle}$ such that the value of the model ETC-pdf decays by a factor a_1 across each discretization interval if $\kappa \in (1, \kappa_3]$, and a_2 if $\kappa \in (\kappa_3, 5]$. For the evolutionary programming algorithm we have chosen $\mu = 16$, or equivalently, a population containing $2\mu = 32$ individual strategies . This corresponds to the splitting of each 256×256 image used in the experiments into 64 sub-blocks B_s of dimensions 32×32 and assigning 2 sub-blocks to each individual regularization strategy. For the purpose of improved accuracy in determining $\hat{E}^{\kappa}_{pdf}(\mathcal{S}_p, \mathcal{R}_p)$, however, we have allowed partial overlapping in the block assignment, resulting in 4 assigned blocks for each individual. In the competition stage of the algorithm, we have employed tournament competition with a tournament size $Q = 10$ as the selection mechanism. To increase the efficiency of the searching process, we have confined the values of $\lambda_{\min}$ within the interval $\mathcal{C}_{\lambda_{\min}} = [0.0005, 0.004]$. The fact that this is not an overly restrictive constraint is evident from Table 7.1 listing the optimal regularization strategies discovered by the current algorithm under a variety of degradation conditions, where we

can see that the optimal values of $\lambda_{\min}$ is well within the above specified interval, and the primary purpose of specifying this constraint is to speed up the searching process. Apart from this, no additional bounds are imposed on the other entries of the regularization strategy to avoid over-constraining the problem. In other words, we have adopted the constraint set $\mathcal{C}_{\mathcal{S}} = \mathcal{C}_{\lambda_{\min}} \times \mathbf{R}^7$ in equation (7.42).

The results for the 5×5 Gaussian PSF with 30dB noise are shown in Figure 7.6 (the degraded image is shown in Figure 3.4(a) in Chapter 3). Figure 7.6(d) shows the restored image using the current evolutionary algorithm. In order to validate the current algorithm as a viable alternative solution of the adaptive regularization problem, we first compare the result with restored images using non-adaptive regularization methods. Figure 7.6(a) shows the under-regularized restored image using the non-adaptive Hopfield neural network restoration approach by Zhou *et.al* [118], with $\lambda = 0.0005$, and Figure 7.6(b) shows the result using $\lambda = 0.004$ to achieve compromise between noise smoothing and feature preservation . In addition, to qualify as a useful adaptive regularization approach, we would expect that the quality of the evolutionary adaptive regularized image should be comparable to our previous result using the non-fuzzy HMBNN approach, which is shown in Figure 7.6(c). For the narrow Gaussian PSF and this moderate level of noise, the results are in general comparable to each other, except in the under-regularized case in Figure 7.6(a) where we can notice noise amplification in the smooth regions.

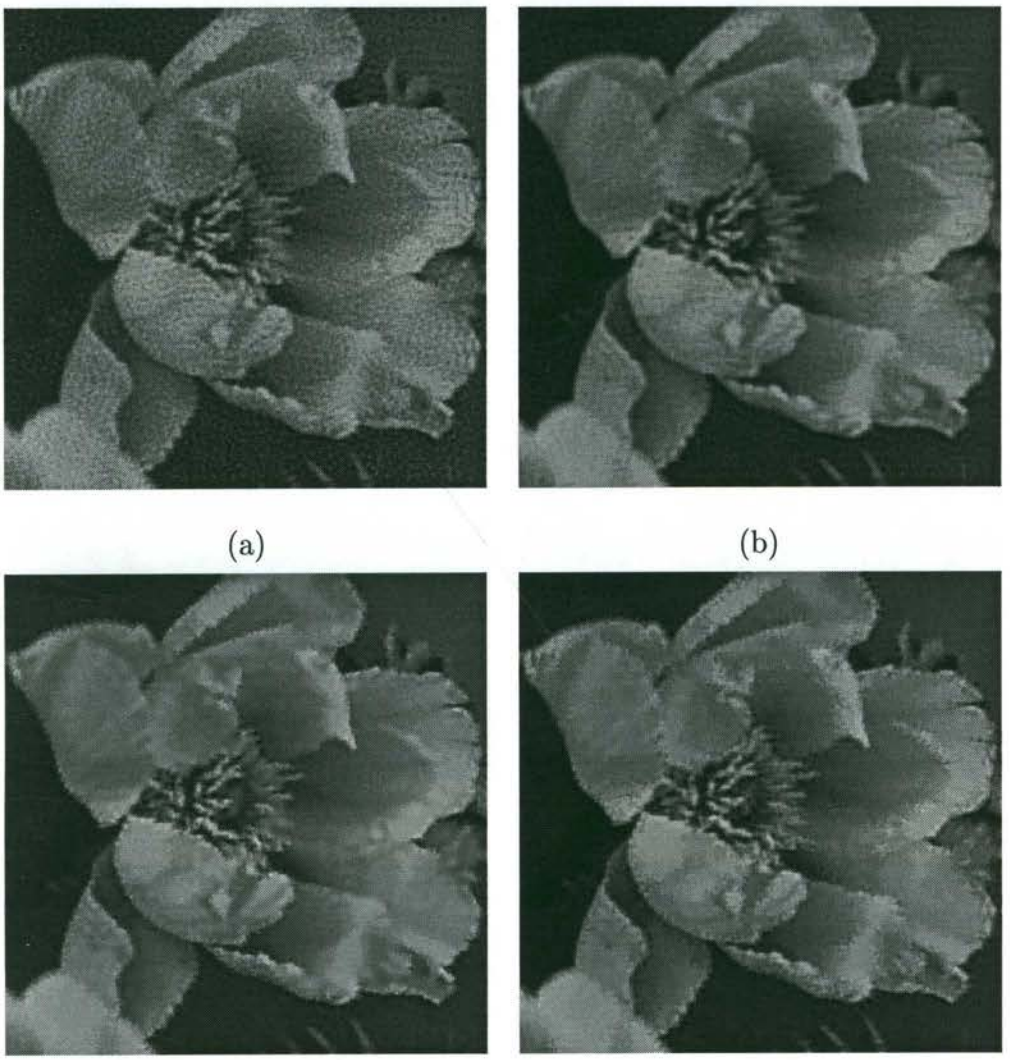
The results for the 5 × 5 uniform PSF with 30dB noise are shown in Figure 7.7 (the degraded image is shown in Figure 3.5(a) in Chapter 3). We can observe that the restored image using the evolutionary approach in Figure 7.7(d) is more preferable compared with the non-adaptive results (Figures 7.7(a) and (b)). We can also see that the current EP result is comparable with our previous non-fuzzy HMBNN result in Figure 7.7(c) even under this increased level of degradation. We can also notice that, in the HMBNN approach, the image is partitioned into connected regions and a single regularization parameter value is applied across each such region. This results in the blurring of some of the textured areas when part of them are classified as smooth regions. On the other hand, the current algorithm assigns a specific regularization parameter value λ_{i_1,i_2} to



(a) (b)

(c) (d)

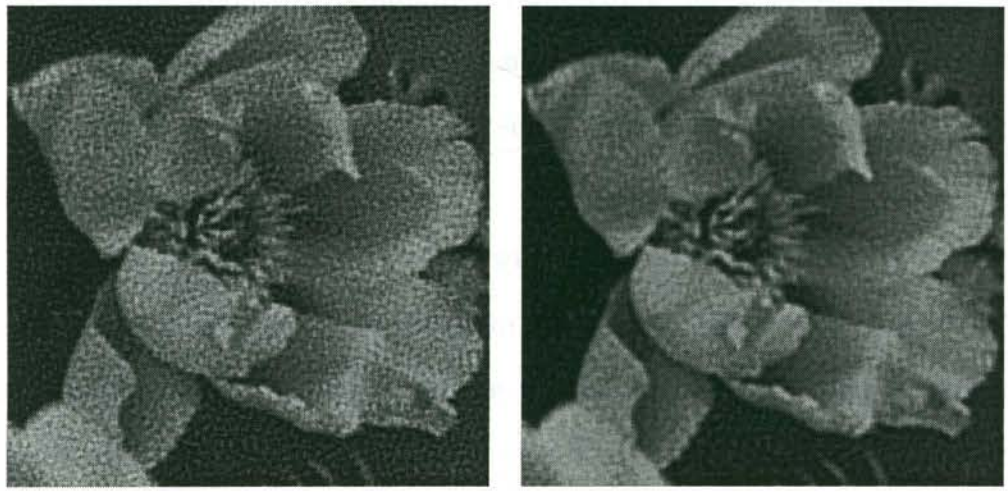
Figure 7.6: Restored flower images (5×5 Gaussian blur, 30dB BSNR). (a)-(d) Restored images using (a) non-adaptive Hopfield restoration algorithm (small λ). (b) non-adaptive Hopfield restoration algorithm (optimally adjusted λ by user) (c) HMBNN. (d) EP.



(a) (b)

(c) (d)

Figure 7.7: Restored flower images (5×5 uniform blur, 30dB BSNR). (a)-(d) Restored images using (a) non-adaptive Hopfield restoration algorithm (small λ). (b) non-adaptive Hopfield restoration algorithm (optimally adjusted λ by user) (c) HMBNN. (d) EP.



(a)

(b)

(c)

(d)

(e)

(f)

Figure 7.8: Restored flower images (5×5 uniform blur, 20dB BSNR). (a)-(f) Restored images using (a) non-adaptive Hopfield restoration algorithm (small λ). (b) non-adaptive Hopfield restoration algorithm (optimally adjusted λ by user) (c) HMBNN. (d) EP (e) Decreased α from EP value (f) Increased α from EP value.

each pixel according to its local standard deviation σ_{i_1,i_2} based on the decreasing sigmoid relationship . As a result, the parameter can adapt itself even within smooth regions to reveal possible hidden textures, and the extent of texture blurring in Figure 7.7(d) is not as serious as in Figure 7.7(c).

The results for the 5×5 uniform PSF with 20dB noise are shown in Figure 7.8 (the degraded image is shown in Figure 3.6(a) in Chapter 3). We notice that the EP result in Figure 7.8(d) is slightly noisier than the non-fuzzy HMBNN result in Figure 7.8(c), especially around the edges. However, compared with the non-adaptive results in Figures 7.8(a) and (b), we can see that the smooth regions in the EP result are correctly regularized. In addition, similar to the non-fuzzy HMBNN result, smooth blotches also appear in the textured regions in Figure 7.8(d). Recall that similar problems occur in the non-fuzzy HMBNN result, it is reasonable to adopt the same set of solutions, i.e, to separately regularize the edges and textures, and to incorporate complementary information regarding the textured regions. These extensions of the current EP algorithm will be discussed later.

Despite these problems, it is seen that the final threshold parameter α adopted by the EP algorithm is appropriate for the current image. To see this, we perturb the evolved threshold parameter α slightly and use the modified strategy to restore the same image in Figures 7.8(e) and (f). In Figure 7.8(e), the parameter α is adjusted such that it is slightly below the evolved value. We can note the resulting noisy appearance of the image due to the misclassification of some of the smooth areas as edge/textures. In Figure 7.8(f), the parameter α is adjusted slightly above the evolved value, which results in the blurring of some of the image features due to their misclassification as smooth regions.

We have also applied this algorithm to other images. Figure 7.9 and 7.10 show the results for Lena and eagle under 5×5 uniform blur at 30dB BSNR (the degraded images for Lena and eagle are shown in Figure 3.9(a) and Figure 3.10(a) in Chapter 3) . In these two cases, the EP results are preferable to the non-adaptive approaches, and comparable to the previous approach based on the non-fuzzy HMBNN.

The λ assignment map for all three images under 5×5 uniform PSF are shown in

Figure 7.9: Restored Lena images (5×5 uniform blur, 30dB BSNR). (a)-(f) Restored images using (a) non-adaptive Hopfield restoration algorithm (small λ). (b) non-adaptive Hopfield restoration algorithm (optimally adjusted λ by user) (c) HMBNN. (d) EP.

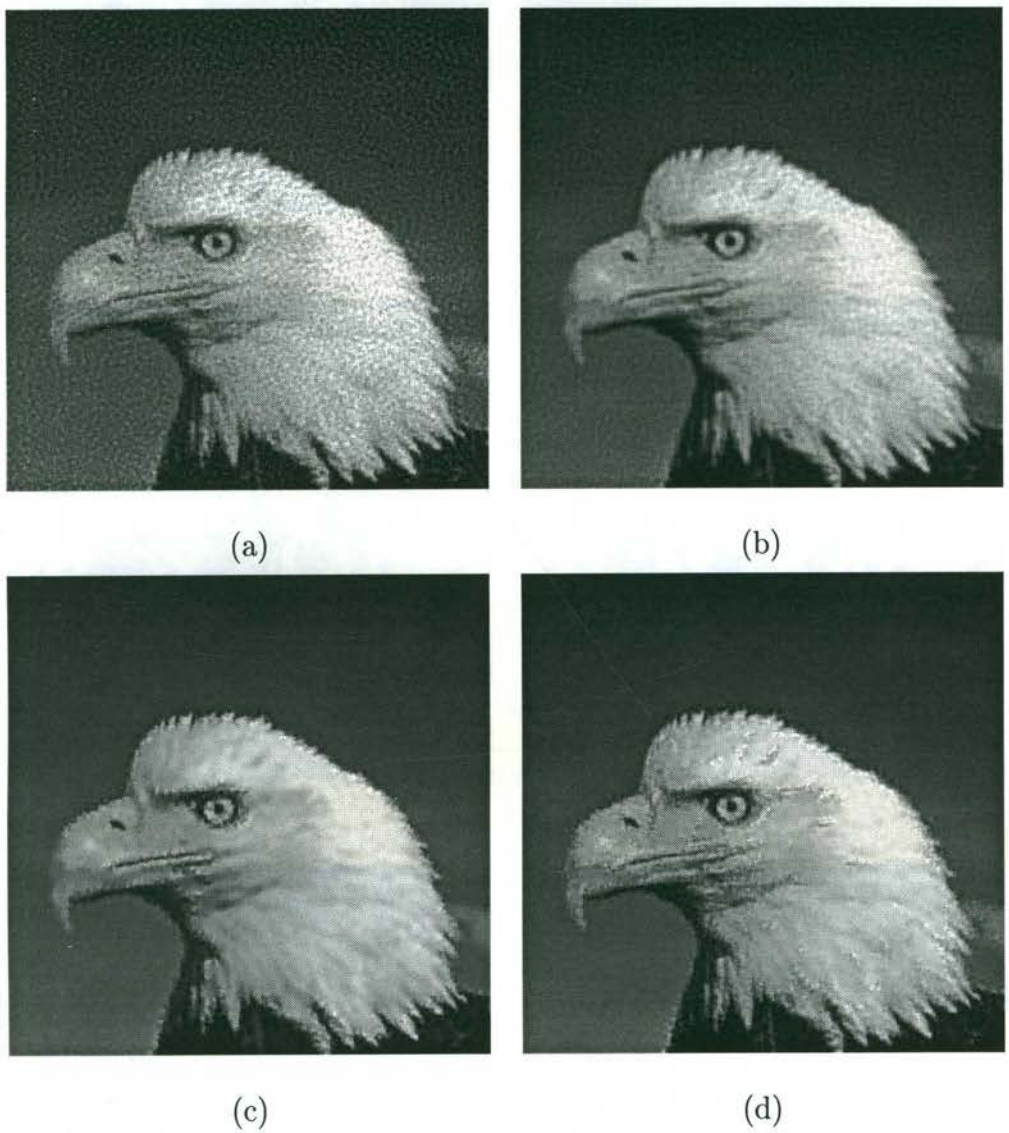
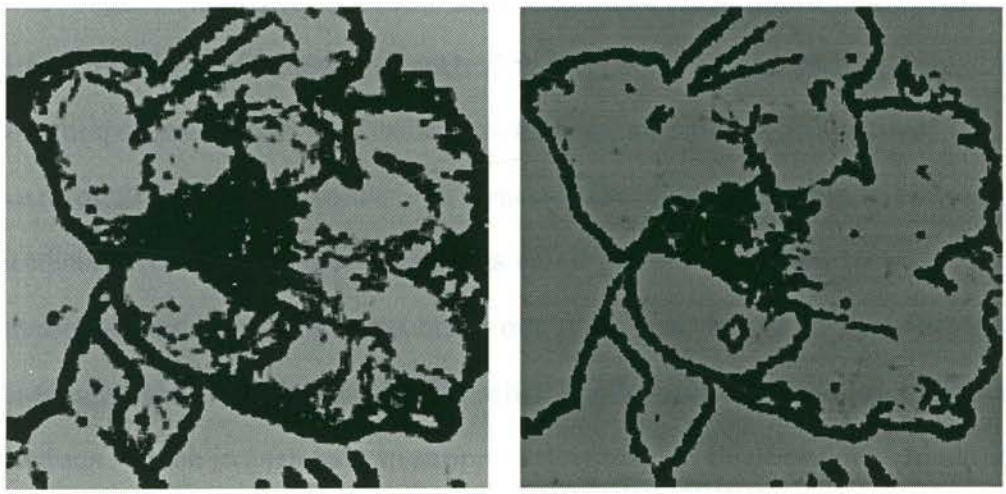
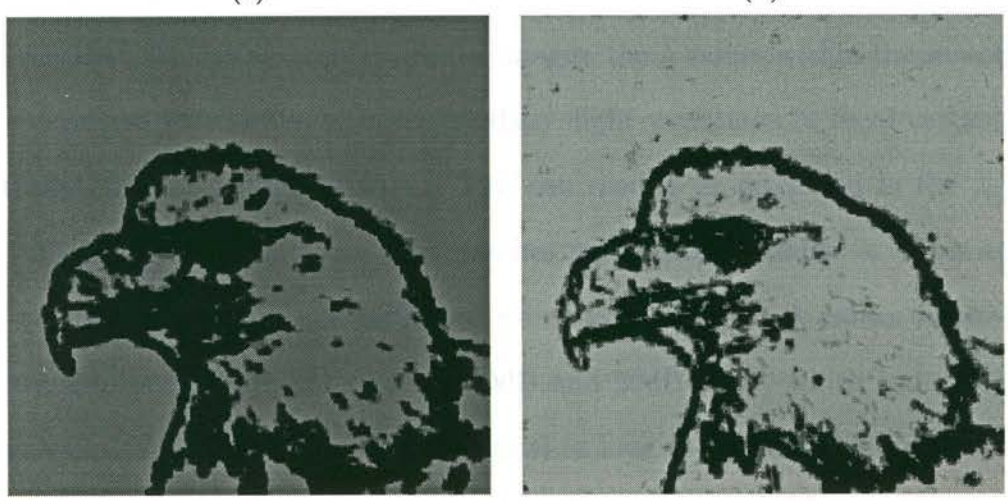


Figure 7.10: Restored eagle images (5×5 uniform blur, 30dB BSNR). (a)-(d) Restored images using (a) non-adaptive Hopfield restoration algorithm (small λ). (b) non-adaptive Hopfield restoration algorithm (optimally adjusted λ by user) (c) HMBNN. (d) EP.



(a) (b)

(c) (d)



(e) (f)

Figure 7.11: The λ-distribution maps for the three images under 5 $\times$5 uniform blur at different levels of additive noise. (a)-(b) Flower (a)30dB BSNR (b) 20dB BSNR (c)-(d) Lena (c) 30dB BSNR (d) 20dB BSNR (e)-(f) Eagle (e) 30dB BSNR (f) 20dB BSNR.

Figure 7.11. Figure 7.11(a),(c),(e) show the assignment maps under 30dB additive noise, and Figure 7.11(b),(d),(f) show the corresponding maps under 20dB noise. In all the assignment maps, the darker gray values correspond to small λ values, and the brighter values correspond to large λ values. In general, the discovered regularization strategy by the artificial evolutionary process assigns small parameter values to textured regions. These smaller values in turn help to bring out more fine details in these regions in the accompanying restoration phase. On the other hand, large λ values are assigned to the smooth regions , which in turn help to suppress the noises in those regions. In addition, the parameters are assigned in such a way that the original structure of the image is respected. The assignment maps for the same image are in general different under different levels of additive noise: for low level of additive noise, the area over which large values of λ are assigned is small compared with the corresponding maps under high level of additive noise. This implies that edge/texture enhancement takes precedence over noise suppression in this case. On the other hand, for higher level of additive noise, most of the areas in the image are assigned large values of λ, and only the very strong edges and texture regions are assigned moderately smaller λ values. We can thus conclude that for low noise levels, the primary purpose is edge/texture enhancement, whereas for higher noise levels it is noise elimination.

It is seen that, for less severely degraded images, the λ values within the smooth image regions can adjust themselves to accommodate slight variations in local variances so as to selectively enhance hidden textures. This can readily be discerned in the λ maps in Figures 7.11 (a),(c) and (e) where gray patches representing smaller λ values appear in some of the smooth regions. In addition, we observe that the λ values in the smooth regions are similar for both the cases of 30dB and 20dB noise, which indicate that the particular value of $\lambda_{\max}$ employed is not critical as long as it is above a certain threshold value for adequate noise suppression.

We also compare the resulting regularization strategies discovered by the artificial evolutionary process under different levels of additive noise . The regularization strategies corresponding to the λ-maps in Figure 7.11 are shown in Table 7.1.

Image, PSF/noise level	$\lambda_{\min}$	$\lambda_{\max}$	α	β
Flower, Uniform PSF/30dB	0.000516	0.0791	8.52	18.09
Flower, Uniform PSF/20dB	0.00119	0.0619	16.40	9.73
Lena, Uniform PSF/30dB	0.000901	0.0931	13.37	17.20
Lena, Uniform PSF/20dB	0.00120	0.0956	18.86	14.59
Eagle, Uniform PSF/30dB	0.000632	0.0555	8.20	13.73
Eagle, Uniform PSF/20dB	0.00198	0.0783	14.69	12.03

Table 7.1: Comparison of the final evolved regularization strategy under different degradation conditions

From Table 7.1 , it is seen that the optimal regularization strategies are in general different for different levels of additive noise. For lower levels of noise, the minimum regularization parameter $\lambda_{\min}$ is seen to be smaller than the corresponding value at higher noise levels. This is reasonable due to the possibility of excessive noise amplification at higher noise levels, which in turn requires higher values of $\lambda_{\min}$ for additional noise suppression.

It is also observed that ,compared with $\lambda_{\min}$, the effect of the maximum parameter value $\lambda_{\max}$ is not as critical under various degradation conditions, provided that it is greater than some threshold value for effectively suppressing the background noise under the particular condition. Suppose that $\lambda^*_{\max}$ corresponds to this value, then any value $\lambda_{\max} > \lambda^*_{\max}$ would be equally effective in suppressing the same level of noise. This explains the values for $\lambda_{\max}$ in Table 7.1 where in some cases, the values for the lower noise levels are even higher than those at higher noise levels.

The parameter α represents the offset of the transition in the sigmoidal regularization function, which implicitly defines the standard deviation threshold separating the small variance from the large variance region. Therefore, this is a critical parameter which determines the overall segmentation of the image into smooth and textured regions. Incorrect choices of this parameter usually result in undesirable artifacts in the final restored

image. For example, if we have chosen this parameter too small, implying that some of the smooth regions are classified as textures, noises will be amplified which will be very visible in the supposedly smooth regions. On the other hand, if we have chosen the parameter too large, some textured regions will be misclassified as smooth regions, which will lead to their excessive blurring. The effect of different choices of this parameter has been illustrated in Figure 7.8(e) and (f).

From Table 7.1, we can observe that α is generally smaller for lower additive noise levels and greater for higher noise levels. This is reasonable in view of the fact that for slight degradations, we can afford using little or no regularization in the textured regions to bring out all the details, even at the expense of misclassifying a very small portion of the smooth regions, because at such slight degradation level, the degree of noise amplification is not very serious and our primary purpose would be edge/texture enhancement. On the other hand, for more severe degradations, we cannot afford to misclassify any smooth regions as textured regions, which will lead to very visible noises. We can, however, afford to misclassify small areas of edge/textured regions even at the expense of smoothing some of the details, as the subjective quality of the resulting image is far more preferable to an otherwise noisy image.

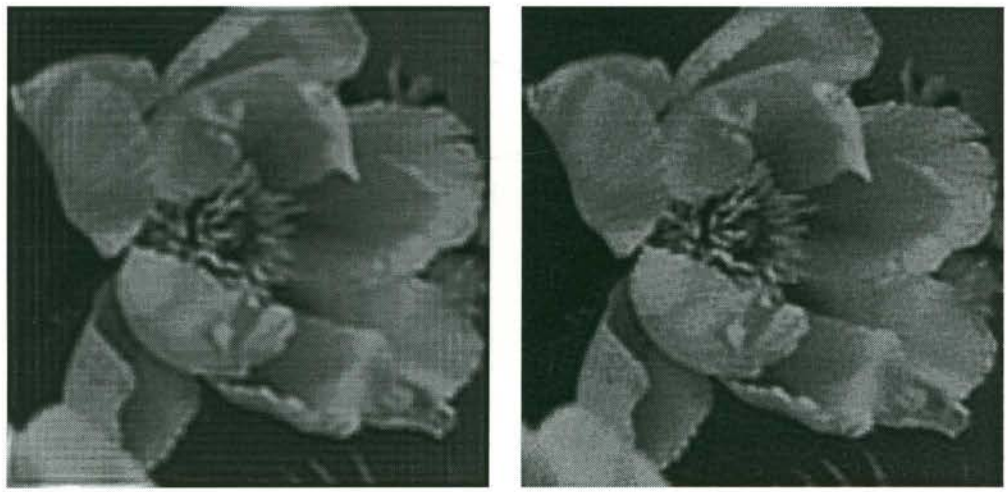
The parameter β controls the steepness of the sigmoidal function transition. The original purpose of using this parameter is to introduce some degree of fuzziness into the adaptive regularization procedure as opposed to the use of hard decisions. Smaller β values correspond to soft decisions in distinguishing between smooth and weak textured regions, which gradually tend towards a hard decision mechanism as β increases. By making this parameter adjustable, we should expect our search algorithm to adopt the optimal level of fuzziness for each picture under different degradation conditions. From the results, it is seen that for the particular images used in our experiments, this parameter is not particularly sensitive to different levels of degradations, and is usually fairly large for most of the cases, indicating that crisp partitions of the image into smooth and edge/textured regions are preferred by the current algorithm.

Although the current algorithm produces satisfactory restoration results even for the

(a) (b)

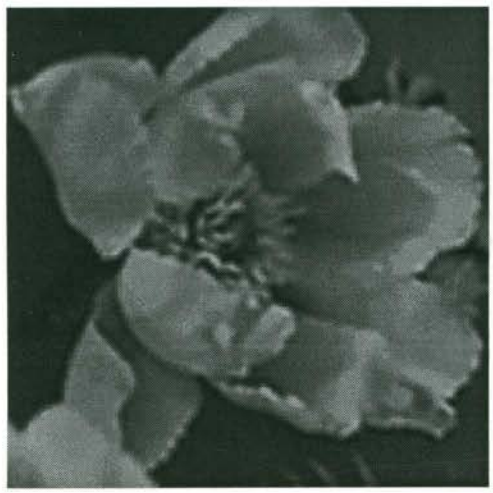
Figure 7.12: Restored images (5 × 5 uniform blur, 20dB BSNR). (a) Lena (b) Eagle

severely degraded flower image in Figure 7.8, similar problems as in the HMBNN algorithm occur when the method is applied to the images Lena and eagle under the same level of degradation (the degraded images for Lena and eagle at 20dB BSNR are shown in Figures 3.12(a) and (b) in Chapter 3). The results are shown in Figure 7.12 under 5 × 5 uniform blur with 20dB noise. It can again be observed that the final evolved λ values in the edge/textured areas are biased in favor of the textured areas, leading to a noisy appearance for the edges. The noise is slightly more visible around the edges compared with the corresponding images for the HMBNN algorithm due to the different cost measures adopted for determining λ. In addition, as in the previous cases, smooth blotches appear in the textured areas of the restored image, which are particularly noticeable within the region containing the feathers in Lena. Recalling the similar types of problem encountered in Chapter 3, it is natural to adopt similar solutions to these problems, namely, the classification of the edge/textured areas into separate edge and textured areas using the ETC measure, and the application of fuzzy criteria in assigning different λ values to the two different areas. The existence of smooth blotches also calls for the inclusion of the texture map, derived using the texture extraction algorithm in Chapter 5, as a complementary input to the restoration algorithm.



(a)

(b)



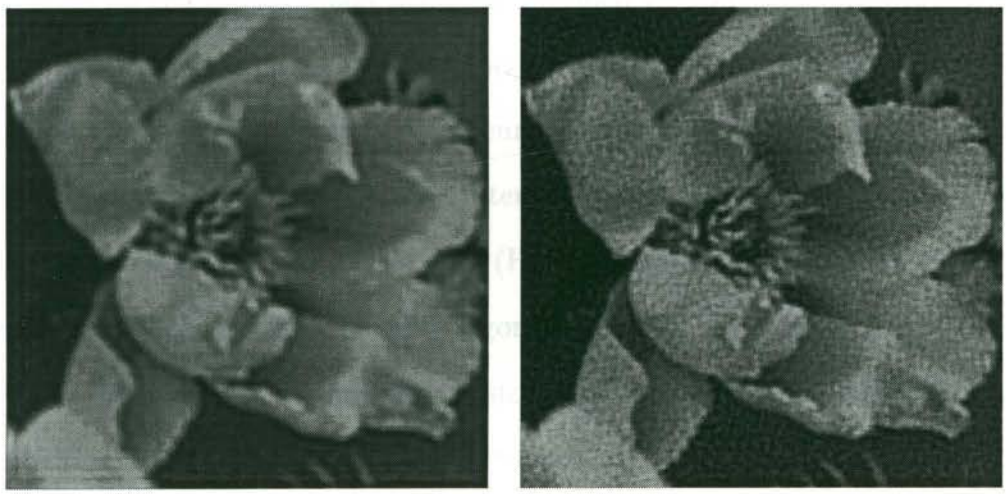
(c)

(d)

(e)

(f)

Figure 7.13: Restored Flower images (5×5 uniform blur, 30dB BSNR). (a)-(f) Restored images using (a) Wiener filter (b) non-adaptive Hopfield restoration algorithm (optimally adjusted λ by user) (c) KK-SAI (d) Non-fuzzy EP (e) Fuzzy HMBNN (f) Fuzzy EP.



(a)

(b)

(c)

(d)

(e)

(f)

Figure 7.14: Restored Flower images (5×5 uniform blur, 20dB BSNR). (a)-(f) Restored images using (a) Wiener filter (b) non-adaptive Hopfield restoration algorithm (optimally adjusted λ by user) (c) KK-SAI (d) Non-fuzzy EP (e) Fuzzy HMBNN (f) Fuzzy EP.

The results for the Flower image using the fuzzified version of the evolutionary restoration algorithm are shown in Figure 7.13, under 5×5 uniform blur at 30dB BSNR. As in Chapter 6, we have compared the the current result in Figure 7.13(f) with those of conventional algorithms such as Wiener filter (Figure 7.13(a)), the Hopfield neural network restoration algorithm by Zhou *et.al* (HNN) (Figure 7.13(b)), and the Kang and Katsaggelos spatially adaptive iterative algorithm (KK-SAI) (Figure 7.13(c)). We have also included the non-fuzzy evolutionary restoration result in Figure 7.13 (d). Due to the only moderate level of degradation, the non-fuzzy result is already satisfactory, but we can notice the slightly noisy appearance of the edges of the petals. This can be compared with the current fuzzy evolutionary result in Figure 7.13(f), where these edge noises are removed without affecting the already near optimally regularized textured regions. We have also included our previous fuzzy HMBNN restoration result in Figure 7.13(e), which represents our previous most satisfactory result. We can see that the current result using the evolutionary programming approach is comparable to the fuzzy HMBNN result.

The difference between the fuzzy and non-fuzzy evolutionary restoration results becomes more apparent when we observe the restored images under 5×5 uniform blur with 20dB noise. In the non-fuzzy EP result in Figure 7.14(d), we can notice the amplified noises in the vicinity of edges. This can be compared with the fuzzy evolutionary result in Figure 7.14(f) where the incorporation of the ETC fuzzy criterion has allowed the selective removal of noises around the edges without affecting the fidelity of the textured areas. This result is also comparable with our previous fuzzy HMBNN result in Figure 7.14(e) under this more severe degradation, indicating that the current evolutionary approach represents a viable alternative solution to the problem of adaptive regularization.

We have also applied the current approach to the Lena and eagle image under 5×5 uniform blur with 20dB noise (Figures 7.15 and 7.16). Conclusions similar to those for the flower image can be drawn regarding these two images: the adoption of the fuzzy criteria in the evolutionary restoration algorithm always results in images with better subjective quality (Figures 7.15(f) and 7.16(f)), and the results are usually comparable to those images restored using the fuzzy HMBNN approach (Figures 7.15(e) and 7.16(e)).

(a)

(b)

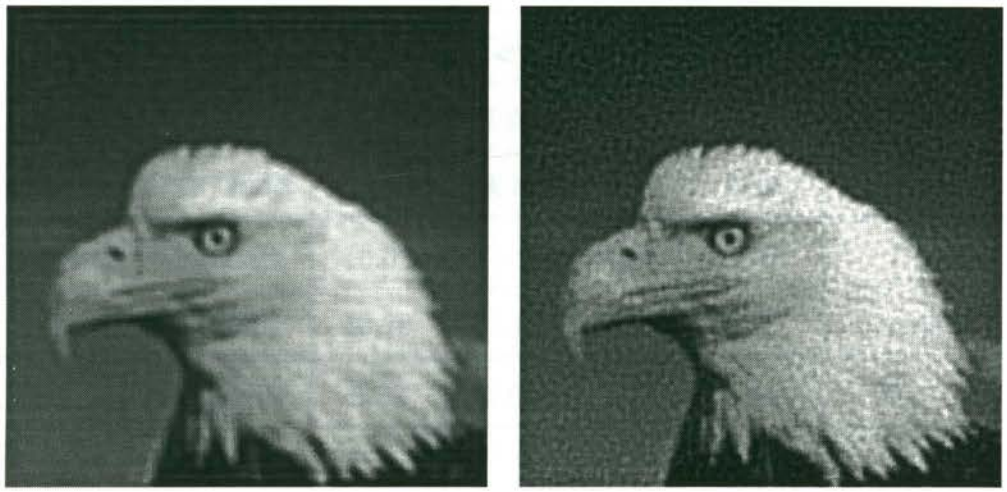
(c)

(d)

(e)

(f)

Figure 7.15: Restored Lena images (5 $\times$ 5 uniform blur, 20dB BSNR). (a)-(f) Restored images using (a) Wiener filter (b) non-adaptive Hopfield restoration algorithm (optimally adjusted λ by user) (c) KK-SAI (d) Non-fuzzy EP (e) Fuzzy HMBNN (f) Fuzzy EP.



(a)

(b)

(c)

(d)

(e)

(f)

Figure 7.16: Restored eagle images (5×5 uniform blur, 20dB BSNR). (a)-(f) Restored images using (a) Wiener filter (b) non-adaptive Hopfield restoration algorithm (optimally adjusted λ by user) (c) KK-SAI (d) Non-fuzzy EP (e) Fuzzy HMBNN (f) Fuzzy EP.

Image, PSF/noise level	Blurred	Wiener	HNN	KK-SAI	F-HMBNN	F-EP
Flower, Uniform PSF/30dB	8.73	7.96	5.76	6.95	5.16	5.48
Flower, Uniform PSF/20dB	10.06	8.12	8.71	8.69	6.94	7.03
Lena, Uniform PSF/30dB	13.03	11.23	8.18	11.15	7.10	7.29
Lena, Uniform PSF/20dB	13.79	11.61	10.67	12.05	9.69	9.65
Eagle, Uniform PSF/30dB	9.88	9.61	7.92	8.64	6.97	7.09
Eagle, Uniform PSF/20dB	11.98	9.75	10.56	11.12	8.99	9.01

Table 7.2: RMSE values of the restoration results using various algorithms

We list the root mean square error (RMSE) of the restored images under the current evolutionary restoration algorithm, together with our previous RMSE values of all other algorithms, in Table 7.2 for ease of comparison (F-HMBNN and F-EP refer to the fuzzy HMBNN and fuzzy EP algorithms respectively). It is seen that, in all the cases, the RMSE values for the current fuzzy EP algorithm are comparable to the values for the fuzzy HMBNN algorithm, and are smaller than the corresponding values of the other conventional algorithms, indicating the effectiveness of the evolutionary approach. The RMSE differences between the NN-based and EC-based algorithms can be attributed to the adoption of different criteria in determining the local regularization parameters.

7.7 Summary

We have proposed an alternative solution to the problem of adaptive regularization in image restoration in the form of an artificial evolutionary algorithm. We first characterize an image by a model discrete probability density function (pdf) of the ETC measure κ, which reflects the degree of correlation around each image pixel, and effectively characterizes smooth regions, textures and edges. An optimally regularized image is thus defined as the one with its corresponding ETC-pdf closest to this model ETC-pdf. In other words, during the restoration process, we have to minimize the difference between the ETC-pdf of

the restored image, which is usually approximated by the ETC-histogram, and the model ETC-pdf. The discrete nature of the ETC-histogram and the non-differentiability of the resulting cost function necessitates the use of evolutionary programming (EP) as our optimization algorithm to minimize the error function. The population-based approach of evolutionary programming provides an efficient method to search for potential optimizers of highly irregular and non-differentiable cost function as the current *ETC-pdf error measure*. In addition, the current problem is also non-stationary, as the optimal regularization strategy in the current iteration of pixel updates is not necessarily the same as the next iteration. The maintenance of a diversity of potential optimizers in the evolutionary approach increases the probability of finding alternative optimizers for the changing cost function. Most significantly, the very adoption of evolutionary programming has allowed us to broaden the range of cost functions in image processing which may be more relevant to the current application, instead of being restricted to differentiable cost functions.

Chapter 8

Conclusions

8.1 Main Conclusions

In this thesis, we have illustrated the essential aspects of the adaptive image processing problem in terms of two applications: the adaptive assignment of the regularization parameters in image restoration, and the adaptive characterization of edges in feature detection applications. These two problems are representative of the general adaptive image processing paradigm in that the three requirements for the solution of this problem: namely the *segmentation* of an image into its main feature types, the *characterization* of each of these features, and the *optimization* of the image model parameters corresponding to the individual features, are present in these two applications. In view of these requirements, we have adopted the three main approaches within the class of *computational intelligence* algorithms, namely neural network techniques, fuzzy set theory, and evolutionary computation, for the solution of the adaptive image processing problem. This is due to the direct correspondence between some of the above requirements with the particular capabilities of specific computational intelligence approaches, which is summarized as follows:

Neural Networks

Although neural networks are usually associated with its capability of optimization, which is amply utilized in the adaptive regularization problem for the optimization of the regional parameters, and in the edge characterization problem for the learning of the edge prototypes, we have also incorporated the capabilities of segmentation and characterization in this class of techniques by adopting a *model-based* neural network with *hierarchical* architecture (HMBNN). The modular architecture of the network results in the division of the total number of neurons into distinct sub-groups, with each sub-group representing a single class of patterns, thus fulfilling the purpose of segmentation. On the other hand, the *model-based* configuration of each neuron enables each neuron sub-group to assimilate the essential properties of its corresponding class of patterns, thus fulfilling the purpose of characterization. In addition, the learning capability of the neural networks allows the adaptive determination of the parameters of the resulting feature model, thus satisfying the purpose of optimization. The importance of these three capabilities associated with this class of specialized neural networks is illustrated by their effectiveness in the adaptive regularization and the adaptive edge characterization problems.

Fuzzy Logic

Fuzzy set techniques are most effective for our requirement of characterization, especially characterization of human preferences with regard to the quality of adaptively processed images. This is due to the inherently ambiguous definition of certain human concepts which cannot be expressed in terms of a crisp classification, but can only be characterized by matters of degrees. A typical example in the context of image processing is the concept of textures and edges, the clarification of which is important in image filtering and restoration applications due to their different noise masking capabilities. This can be facilitated by adopting a fuzzy characterization in terms of the degree of resemblance of a particular gray level configuration to either textures or edges.

Evolutionary Computation

Evolutionary computation represents a powerful optimization approach which readily satisfies the corresponding requirement on the part of adaptive image processing. Due to its stochastic nature, which reduces the instances of local minima, it is especially effective for non-convex, and even non-differentiable cost functions specifying the requirements of a particular image processing operation. This is in contrast with neural network techniques based on gradient learning algorithms which is more susceptible to spurious local minima in the former non-convex case, and is altogether not applicable in the latter non-differentiable case. An important implication of adopting this class of techniques is the possible broadening of the class of cost functions applicable to image processing, which can possibly reflect the nature of the current problem more accurately, instead of being restricted to differentiable but non-optimal cost functions.

Through our application of the above computational intelligence techniques to the problem of adaptive image processing applications, we can draw the following conclusions with regard to the merit of this approach:

8.1.1 Effectiveness of adopting a learning approach in adaptive regularization

Instead of using a trial and error approach in selecting the regularization parameters in image restoration, we have adopted a *learning* approach for this task, with the regularization parameters in various local regions of the image acting as network weights of a *model-based* neural network. The additional incorporation of a modular architecture for the network allows the adoption of different local image models for individual regions, and the regional regularization parameters are then adjusted using a gradient descent learning algorithm in accordance with the corresponding local model. This is in contrast with conventional approaches where the parameters are determined solely as functions of the partially restored image, such that the current learning approach offers additional flexibility in determining the final image quality by judiciously choosing the various im-

age models. More importantly, once the local image models are specified, the current approach can be applied to images with various degrees of degradations through the automatic adjustment of the regularization parameters, without further need of re-specifying the original models

From the experimental results, it is seen that the current approach has correctly assigned different parameter values to different image feature types, with small parameter values being assigned to the edges and textured regions for detail enhancement, and large values being assigned to the smooth regions for noise suppression, resulting in a pleasing appearance of the restored images. The results are in general superior to those restored using a non-adaptive approach. More importantly, the network automatically adjusts the various regularization parameters to accommodate different degradation conditions, thus demonstrating the effectiveness of the current approach.

8.1.2 Robustness of neural network-based edge characterization

The previously described model-based neural network with hierarchical architecture is also suitable for the task of image *edge characterization*, where the various sub-networks of the overall network encode the different edge prototypes existing under different levels of background illumination. This is important in view of the different human preferences in regarding certain discontinuities in gray level values as significant features under different illumination conditions. More importantly, the representation of the edge prototypes as network weights allow their automatic adjustment using a learning process. Instead of communicating their preferences through the specification of threshold parameter values to edge detection algorithms, human users can now highlight specific edge examples on a particular image, and the corresponding pixel configurations are then incorporated as training examples for the neural network.

Our experiments have shown that the trained network is capable of generalizing from the sparse examples supplied by the human user to identify all the important edges, both on images from which the training data originate, and on novel images. Most importantly,

the trained network can be directly applied, without further re-training and architecture alteration, to noisy images and still identify the correct edges, thus demonstrating the robustness of the current approach. This is unlike conventional edge detection approaches where the thresholds are usually required to be re-adjusted under similar circumstances.

8.1.3 Formulation of the Edge-Texture Characterization (ETC) measure

The question of whether it is possible to distinguish between edges and textures without involving extensive computation is answered in the affirmative by the formulation of the Edge-Texture Characterization (ETC) Measure. Unlike the usual measures of image activities such as local variance or gradient magnitude which produce similar values for edges and textures, and thus cannot distinguish between these two, the ETC measure assigns different *scalar* values to specific pixel configurations with different correlational properties. As a result, edges and textures are usually assigned different ETC measure values due to their different underlying pixel configurations. More importantly, we can analytically establish intervals of measure values corresponding to those pixel configurations which resemble edges more than textures, and *vice versa*. This is confirmed by experiments where we can reasonably isolate the edges and textures of various image by highlighting those pixels with measure values within the above intervals.

8.1.4 Incorporation of edge-texture discrimination in adaptive regularization

The importance of edge-texture discrimination in image restoration is highlighted in our previous experiments using an HMBNN for adaptive regularization, where the unsuitability of imposing similar levels of regularization for both edges and textures was noted due to the different noise masking capabilities of these two feature types. The newly formulated ETC measure thus serves as an ideal candidate for this task. The continuous transition between edge and texture pixel configurations with variations of this measure

value, together with the inherent ambiguities associated with the words "edge" and "texture", naturally suggests the formulation of a fuzzy model where these two concepts are embodied in two corresponding fuzzy sets defined on the domain of ETC measure. At the same time, the previous sub-network structure of the HMBNN is extended to include two neurons, each of which computes the relevant variables for restoration under the assumption that the underlying pixel configuration corresponds to edges or textures. The output of these two neurons are then combined in an augmented *output layer* to give the final gray level update value, with the edge/texture fuzzy membership values of the current pixel configuration serving as output weights for the network. Experiments have shown that this capability of distinguishing between edges and textures greatly improve the overall appearances of the restored images, especially for images under more severe degradations, compared with our previous results.

8.1.5 Effectiveness of evolutionary parameterization in image restoration

We have also explored an alternative solution to the problem of adaptive regularization using evolutionary programming. This is due to our recognition of the similar form for the pdf (probability density function) of the ETC measure values for a large class of images, which thus serves as a *signature* for a large number of non-degraded images. On the other hand, the corresponding form of the ETC-pdf for degraded images are usually very different from the non-degraded ETC-pdf. Our purpose, therefore, is to restore the image in such a way that its ETC-pdf once again conforms to the non-degraded ETC-pdf.

The necessity to approximate the ETC-pdf of real images using histograms, and the resulting non-differentiability of the associated cost function, requires the adoption of powerful optimization techniques of which evolutionary programming is a prominent example. We have specified the individual optimizers in the population in such a way as to include essential regularization and segmentation parameters, so that the successful completion of the optimization process will result in the simultaneous performances of adaptive reg-

ularization and segmentation. This is in contrast with the previous model-based neural network approach where a preliminary segmentation of the image is required.

Experiments have indicated that the evolutionary approach produced restored images with qualities fully comparable with those restored using the previous HMBNN approach. Besides confirming the current approach as a viable alternative solution to the problem of adaptive regularization in image restoration, a more important implication is the possibility of extending this evolutionary paradigm to other image processing applications such that more accurate cost functions specifying the operations of these applications can be incorporated, instead of being restricted to cost functions suitable only for gradient-based optimization.

8.2 Suggestions for Future Works

8.2.1 Improvements for Current Works

Image Restoration

The primary focus of this thesis is on image restoration, and in particular the adaptive assignment of the associated regularization parameters for optimal visual results. This is one of the possible approaches in locally regulating the quality of the restored image, wherein a quadratic function is adopted as the regularization functional, and different regularization parameter values are assigned to specific spatial locations. Another possible approach is to adopt a non-quadratic cost function [18] which is also characterized by a set of parameters which determine the degree of smoothness for the resulting image. In other words, the task of determining those parameters is similar to our current task of determining the regularization parameters, but the effect of varying the parameters of the non-quadratic functional on the visual quality of the image may be very different from that of varying the regularization parameters. As a result, a possible future research direction would be to adapt our current HMBNN approach such that the network weights represent the parameters characterizing these functionals, and to compare the corresponding results

with the current case using the quadratic regularization functional.

Another potential research area is the application of the current approach to cases where the point spread function (PSF) is only partially specified or even totally unspecified [63] . In the current work, we assume that we have complete knowledge of the point spread function, but in certain cases the measurement of the PSF may be subject to error, or its form is altogether unknown. In these cases, we have to resort to the process of blind or partially blind deconvolution [63] where the parameters of the PSF, together with the regularization parameters, have to be estimated using the information from the degraded image alone. The solution to this problem may require the design of a special class of sub-network in our HMBNN architecture where the associated weights represent the unknown PSF coefficients.

Still another possible application area is in the restoration of images subject to spatially variant degradations [80, 104] . As opposed to the specification of spatially varying regularization parameters, the specification of spatially varying PSF represents a far more difficult problem due to the much higher sensitivity of the quality of the restored image to errors in the PSF. Assuming the possible partitioning of the different PSFs into classes, an HMBNN can be designed such that each sub-network can be used to estimate the coefficients of a single PSF class.

General Feature Characterization and Detection

In this thesis, we have described a neural network-based approach for edge characterization and detection, where human-specified edge examples are used to establish edge prototypes within the network through the process of training. A natural generalization of this approach is to incorporate other types of image features which human beings would usually regard as significant. These include textures, lines, corners or possibly even higher-level scene descriptions. In addition to the designated sub-networks for the edge prototypes, we can establish different classes of sub-networks for texture prototypes, line prototypes, or corner prototypes respectively. Similar to the case for edge characterization, we can ask human users to highlight their preferred examples corresponding to each of the above

features on an image, and then we can incorporate these examples as training data for the respective sub-networks for prototype establishment. It is anticipated that the ETC measure will play an important role here in the characterization of features.

8.2.2 Extensions to other application areas

Although our primary focus in this thesis is on the problem of image restoration and feature characterization using computational intelligence techniques, these two applications are representative of the typical requirements for formulating adaptive image processing algorithms, and the generality of the approaches adopted can be readily extended to other applications. In particular, in contrast with the requirement of optimization which is comparatively more application-dependent, the requirements of segmentation and characterization of individual image feature types have to be addressed in a wide variety of image processing applications. Some of the possible extensions of our current line of work include

Adaptive image filtering

Many of the problems encountered in adaptive image filtering [5, 87] is similar to those in image restoration, and in most of the literatures the image filtering problem is also classified under the topic of image restoration. The primary difference between the filtering and our current deblurring operation is that, while the degradation in the latter involves mostly blurring and moderate levels of noises, the former usually involves very high levels of noises of possibly mixed statistical properties, but with little blurring [87]. In contrast with the deblurring operations which are important in astronomical and medical image processing [55, 67] , the filtering operation is more prevalent in video processing applications [20] where Gaussian noises and impulse-like noises with heavy-tailed distributions co-exist.

Many of the criteria in adaptive regularization is thus also applicable to adaptive filtering. For example, increased levels of smoothing is usually required for the smooth regions

due to the more visible noises in those regions, and less smoothing can be applied around the edges and textures, although the specific mechanism for implementing this adaptive filtering operation is distinctly different from that in deblurring, which would thus require modification and possibly extension of our current computational intelligence approach. The newly formulated ETC measure is also highly relevant for this alternative application: due to their different noise masking capabilities, it is also desirable to differentiate between edge and texture in adaptive filtering applications, such that different levels of filtering can be applied to each of them.

Adaptive image compression

The segmentation and characterization of individual image feature types such as edges, textures, and smooth regions are also important for the application of adaptive image compression [91, 99, 106] , where separate coding strategies are applied to each of the above three features to minimize the overall compression ratio. Modification of our current computational intelligence approach can thus be applied to segment an image into its constituent components, and then characterize the individual features in terms of appropriate image models. A final optimization process will then adjust the parameters of individual image models so as to minimize the entropy of the residual variations unaccounted for by the models. The resulting coding gain can be expected to be substantial over non-adaptive image compression, due to the need to transmit only the model parameters and the residual sequence.

Bibliography

[1] R. Anand, K. Mehrotra, C. K. Mohan, and S. Ranka, "Efficient classification for multiclass problems using modular neural networks," *IEEE Trans. Neural Networks*, vol. 6, no. 1, pp. 117–124, 1995.

[2] J. A. Anderson, *An Introduction to Neural Networks.* Cambridge, MA.: MIT Press, 1995.

[3] H. Andrews and B. Hunt, *Digital Image Restoration.* Englewood Cliffs, NJ: Prentice Hall, 1977.

[4] N. Ansari and E. Hou, *Computational Intelligence for Optimization.* Boston: Kluwer Academic, 1997.

[5] G. Arce and R. Foster, "Detail-preserving ranked-order based filters for image processing," *IEEE Trans. Acoust., Speech, Signal Processing*, vol. 37, no. 1, pp. 83–98, 1989.

[6] E. Ardizzone, A. Chella, and R. Pirrone, "A neural based approach to image segmentation," in *Proc. Int. Conf. on Artificial Neural Networks (ICANN'94)*, pp. 1153–1156, 1994.

[7] T. Bäck, *Evolutionary Algorithms in Theory and Practice.* New York: Oxford Univ. Press, 1996.

[8] T. Bäck, D. B. Fogel, and Z. Michalewicz, eds., *Handbook of Evolutionary Computation.* New York: Oxford Univ. Press and Institute of Physics, 1997.

[9] T. Bäck, U. Hammel, and H.-P. Schwefel, "Evolutionary computation: Comments on the history and current state," *IEEE Trans. Evolutionary Comp.*, vol. 1, no. 1, pp. 3–17, 1997.

[10] T. Bäck and H.-P. Schwefel, "Evolutionary computation: An overview," in *Proc. 3rd IEEE Conf. on Evolutionary Computation*, (Piscataway, NJ), pp. 20–29, IEEE Press, 1996.

[11] P. Baldi and K. Hornik, "Neural networks and principal component analysis: learning from examples without local minima," *Neural Networks*, vol. 2, no. 1, pp. 53–58, 1989.

[12] M. R. Banham and A. K. Katsaggelos, "Digital image restoration," *IEEE Signal Processing Magazine*, vol. 14, no. 2, pp. 24–41, 1997.

[13] M. Barnsley, *Fractals Everywhere.* New York, NY: Academic Press, 1988.

[14] M. Bertero, T. Poggio, and V. Torre, "Ill-posed problems in early vision," *Proc. IEEE*, vol. 76, no. 8, pp. 869–889, 1988.

[15] J. C. Bezdek, *Pattern Recognition with Fuzzy Objective Function Algorithms.* New York, NY.: Plenum Press, 1981.

[16] J. C. Bezdek, R. Chandrasekhar, and Y. Attikiouzel, "A geometric approach to edge detection," *IEEE Trans. Fuzzy Systems*, vol. 6, no. 1, pp. 52–75, 1998.

[17] J. C. Bezdek and D. Kerr, "Training edge detecting neural networks with model-based examples," in *Proc. 3rd IEEE Int. Conf. Fuzzy Syst.*, (Piscataway, NJ), pp. 894–901, 1994.

[18] C. Bouman and K. Sauer, "A generalized Gaussian image model for edge-preserving MAP estimation," *IEEE Trans. Image Processing*, vol. 2, no. 3, pp. 296–310, 1993.

[19] A. C. Bovik, T. S. Huang, and D. C. Munson, "A generalization of median filtering using linear combinations of order statistics," *IEEE Trans. Acoust., Speech, Signal Processing*, vol. 31, no. 6, pp. 1342–1350, 1983.

[20] J. C. Brailean, R. P. Kleihorst, S. Efstratiadis, A. K. Katsaggelos, and R. L. Lagendijk, "Noise reduction filters for dynamic image sequences: a review," *Proc. IEEE*, vol. 83, no. 9, pp. 1272–1292, 1995.

[21] T. Caelli, D. Squire, and T. Wild, "Model-based neural networks," *Neural Networks*, vol. 6, no. 5, pp. 613–625, 1993.

[22] J. Canny, "A computational approach to edge detection," *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. 8, no. 6, pp. 679–698, 1986.

[23] R. Chellappa and S. Chatterjee, "Classification of textures using Gaussian Markov random fields," *IEEE Trans. Acoust., Speech, Signal Processing*, vol. 33, no. 4, pp. 959–963, 1985.

[24] R. Chellappa and R. L. Kashyap, "Digital image restoration using spatial interaction models," *IEEE Trans. Acoust., Speech, Signal Processing*, vol. 30, no. 6, pp. 461–472, 1982.

[25] Y. S. Choi and R. Krishnapuram, "A robust approach to image enhancement based on fuzzy logic," *IEEE Trans. Image Processing*, vol. 6, no. 6, pp. 808–825, 1997.

[26] G. R. Cross and A. K. Jain, "Markov random field texture models," *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. 5, no. 1, pp. 25–39, 1983.

[27] H. Derin and H. Elliott, "Modelling and segmentation of noisy and textured images using Gibbs random fields," *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. 9, no. 1, pp. 39–55, 1987.

[28] N. Fang and M. C. Cheng, "An automatic crossover point selection technique for image enhancement using fuzzy sets," *Pattern Recognition Letters*, vol. 14, no. 5, pp. 397–406, 1993.

[29] D. B. Fogel, *Evolutionary Computation: Toward a New Philosophy of Machine Intelligence.* Piscataway, NJ: IEEE Press, 1995.

[30] L. J. Fogel, A. J. Owens, and M. J. Walsh, *Artificial Intelligence Through Simulated Evolution.* New York: Wiley, 1966.

[31] H. Fu and Y. Y. Xu, "Multilinguistic handwritten character recognition by Bayesian decision-based neural networks," *IEEE Trans. Signal Proc.*, vol. 46, no. 10, pp. 2781–2789, 1998.

[32] N. P. Galatsanos and A. K. Katsaggelos, "Methods for choosing the regularization parameter and estimating the noise variance in image restoration and their relation," *IEEE Trans. Image Processing*, vol. 1, no. 3, pp. 322–336, 1992.

[33] S. Geman and D. Geman, "Stochastic relaxation, Gibbs distribution, and the Bayesian restoration of images," *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. 6, no. 6, pp. 721–741, 1984.

[34] A. Ghosh, N. R. Pal, and S. K. Pal, "Image segmentation using a neural network," *Biological Cybernetics*, vol. 66, no. 2, pp. 151–158, 1991.

[35] D. E. Goldberg, *Genetic Algorithms in Search, Optimization and Machine Learning.* Reading, MA: Addison-Wesley, 1989.

[36] R. C. Gonzalez and R. Woods, *Digital Image Processing.* Reading, Massachusetts: Addison-Wesley, 1992.

[37] L. Guan, "Model-based neural evaluation and iterative gradient optimization in image restoration and statistical filtering," *Journal of Electronic Imaging*, vol. 4, no. 4, pp. 407–412, 1995.

[38] B. L. M. Happel and J. M. J. Murre, "Design and evolution of modular neural network architectures," *Neural Networks*, vol. 7, no. 6-7, pp. 985–1004, 1994.

[39] R. M. Haralick, "Digital step edges from zero crossings of second directional derivatives," *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. 6, no. 1, pp. 58–68, 1984.

[40] R. J. Hathaway and J. C. Bezdek, "Local convergence of the fuzzy c-means algorithms," *Pattern Recognition*, vol. 19, no. 6, pp. 477–480, 1986.

[41] S. Haykin, *Neural Networks: A Comprehensive Foundation.* New York: Macmillan, 1994.

[42] J. Hertz, A. Krogh, and R. G. Palmer, *Introduction to the Theory of Neural Computation.* Santa Fe Institute, Addison-Wesley, 1991.

[43] J. H. Holland, *Adaptation in Natural and Artificial Systems.* Ann Arbor, MI: Univ. of Michigan Press, 1975.

[44] J. Hopfield and D. Tank, "Neural computation of decisions in optimization problems," *Biol. Cybern.*, vol. 52, no. 3, pp. 141–152, 1985.

[45] J. Hsieh, "Image enhancement with a fuzzy logic approach," *Electronic Letters*, vol. 31, no. 9, pp. 708–710, 1995.

[46] C. L. Huang, "Parallel image segmentation using modified Hopfield network," *Pattern Recognition Letters*, vol. 13, no. 5, pp. 345–353, 1992.

[47] B. R. Hunt, "The application of constrained least squares estimation to image restoration by digital computers," *IEEE Trans. Comput.*, vol. 22, no. 9, pp. 805–812, 1973.

[48] R. A. Jacobs and M. I. Jordan, "Learning piecewise control strategies in a modular neural network architecture," *IEEE Trans. Syst. Man Cybern.*, vol. 23, no. 2, pp. 337–345, 1993.

[49] A. K. Jain, "Advances in mathematical models for image processing," *Proc. IEEE*, vol. 69, no. 5, pp. 502–528, 1981.

[50] A. K. Jain, *Fundamentals of Digital Image Processing.* Englewood Cliffs, NJ: Prentice Hall, 1989.

[51] J. S. R. Jang, C. T. Sun, and E. Mizutani, *Neuro-Fuzzy and Soft Computing: a Computational Approach to Learning and Machine Intelligence.* Upper Saddle River, NJ: Prentice Hall, 1997.

[52] E. R. Kandel and J. H. Schwartz, *Principles of Neural Science.* New York, NY.: Elsevier, 1985.

[53] M. G. Kang and A. K. Katsaggelos, "General choice of the regularization functional in regularized image restoration," *IEEE Trans. Image Processing*, vol. 4, no. 5, pp. 594–602, 1995.

[54] R. L. Kashyap and R. Chellappa, "Estimation and choice of neighbors in spatial interaction models of images," *IEEE Trans. Inform. Theory*, vol. 29, no. 1, pp. 60–72, 1983.

[55] A. K. Katsaggelos, ed., *Digital Image Restoration.* Berlin, Germany: Springer Verlag, 1991.

[56] A. K. Katsaggelos and M. G. Kang, "Spatially adaptive iterative algorithm for the restoration of astronomical images," *Int. J. of Imaging Systems and Technology*, vol. 6, no. 4, pp. 305–313, 1995.

[57] A. Katsaggelos and M. G. Kang, "Iterative evaluation of the regularization parameter in regularized image restoration," *Journal of Visual Comm. and Image Rep.*, vol. 3, no. 4, pp. 446–455, 1992.

[58] A. Kehagias and V. Petridis, "Time-series segmentation using predictive modular neural networks," *Neural Computation*, vol. 9, no. 8, pp. 1691–1709, 1997.

[59] G. J. Klir and T. A. Folger, *Fuzzy Sets, Uncertainty and Information.* Englewood Cliffs, NJ.: Prentice Hall, 1988.

[60] T. Kohonen, *Self-Organizing Maps.* Berlin: Springer-Verlag, 2nd ed., 1997.

[61] B. Kosko, *Neural Networks and Fuzzy Systems.* Englewood Cliffs, NJ.: Prentice Hall, 1992.

[62] R. Krishnapuram, J. M. Keller, and Y. Ma, "Qnantitative analysis of properties and spatial relations of fuzzy image regions," *IEEE Trans. Fuzzy Systems*, vol. 1, no. 3, pp. 222–33, 1993.

[63] D. Kundur and D. Hatzinakos, "Blind image deconvolution," *IEEE Signal Processing Magazine*, vol. 13, no. 3, pp. 43–64, 1996.

[64] S. Y. Kung, M. Fang, S. P. Liou, M. Y. Chiu, and J. S. Taur, "Decision-based neural network for face recognition system," in *Proc. Int. Conf. on Image Processing*, pp. 430–433, 1995.

[65] S. Y. Kung and J. S. Taur, "Decision-based neural networks with signal/image classification applications," *IEEE Trans. Neural Networks*, vol. 6, no. 1, pp. 170–181, 1995.

[66] O. J. Kwon and R. Chellappa, "Region adaptive subband image coding," *IEEE Trans. Image Processing*, vol. 7, no. 5, pp. 632–48, 1998.

[67] R. Lagendijk and J. Biemond, *Iterative Identification and Restoration of Images.* Boston: Kluwer Academic Publishers, 1991.

[68] R. L. Lagendijk, J. Biemond, and D. E. Boekee, "Regularized iterative image restoration with ringing reduction," *IEEE Trans. Acoust., Speech, Signal Processing*, vol. 36, no. 12, pp. 1874–1888, 1988.

[69] S. H. Lin and S. Y. Kung, "Probabilistic DBNN via expectation-maximization with multi-sensor classification applications," in *Proc. Int. Conf. on Image Processing*, pp. 236–239, 1995.

[70] S. H. Lin, S. Y. Kung, and L. J. Lin, "Face recognition/detection by probabilistic decision-based neural networks," *IEEE Trans. Neural Networks*, vol. 8, no. 1, pp. 114–132, 1997.

[71] D. G. Luenberger, *Linear and Nonlinear Programming.* Reading, MA., Addison-Wesley, 1986.

[72] J. MacQueen, "Some methods for classification and analysis of multivariate observations," in *Proc. 5th Berkeley Symp. on Math. Stat. and Prob.*, vol. 1, pp. 281–296, 1967.

[73] B. Mandelbrot, *Fractals and a New Geometry of Nature.* San Francisco, CA: Freeman, 1981.

[74] D. Marr and E. Hildreth, "Theory of edge detection," *Proc. Royal Soc.*, vol. B-207, pp. 187–217, 1980.

[75] M. Mitchell, *An Introduction to Genetic Algorithms.* Cambridge MA: MIT Press, 1996.

[76] E. Oja, "A simplified neuron model as a principal component analyzer," *Journal of Mathematical Biology*, vol. 15, no. 3, pp. 267–273, 1982.

[77] E. Oja, "Neural networks, principal components and subspaces," *Int. J. of Neural Systems*, vol. 1, no. 1, pp. 61–68, 1989.

[78] S. Osher and L. I. Rudin, "Feature-oriented image enhancement using shock filters," *SIAM Journal on Numerical Analysis*, vol. 27, no. 4, pp. 919–940, 1990.

[79] S. Ozawa, K. Tsutsumi, and N. Baba, "An artificial modular neural network and its basic dynamical characteristics," *Biological Cybernetics*, vol. 78, no. 1, pp. 19–36, 1998.

[80] M. K. Ozkan, A. M. Tekalp, and M. I. Sezan, "POCS-based restoration of space-varying blurred images," *IEEE Trans. Image Processing*, vol. 3, no. 4, pp. 450–454, 1994.

[81] J. K. Paik and A. K. Katsaggelos, "Image restoration using a modified Hopfield network," *IEEE Trans. Image Processing*, vol. 1, no. 1, pp. 49–63, 1992.

[82] N. R. Pal and J. C. Bezdek, "On cluster validity for the fuzzy c-means model," *IEEE Trans. Fuzzy Systems*, vol. 3, no. 3, pp. 370–379, 1995.

[83] W. Pedrycz, *Computational Intelligence: an Introduction.* Boca Raton, FL.: CRC Press, 1997.

[84] P. Perona and J. Malik, "Scale-space and edge detection using anisotropic diffusion," *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. 12, no. 7, pp. 629–639, 1990.

[85] S. Perry and L. Guan, "Image restoration using a neural network with an adaptive constraint factor," in *Proc. IEEE CESA'96 IMACS Multiconference*, pp. 854–859, 1996.

[86] C. Peterson and B. Söderberg, "A new method for mapping optimization problems onto neural networks," *Int. J. of Neural Systems*, vol. 1, no. 1, pp. 3–22, 1989.

[87] I. Pitas and A. Venetsanopoulos, *Nonlinear Digital Filters : Principles and Applications.* Boston: Kluwer Academic Publishers, 1990.

[88] T. Poggio, V. Torre, and C. Koch, "Computational vision and regularization theory," *Nature*, vol. 317, no. 6035, pp. 314–319, 1985.

[89] Proceedings, *IEEE World Congress on Computational Intelligence.* New York, NY.: IEEE Press, 1998.

[90] W. Qian and L. P. Clarke, "Wavelet-based neural network with fuzzy-logic adaptivity for nuclear image restoration," *Proc. IEEE*, vol. 84, no. 10, pp. 1458–1473, 1996.

[91] H. R. Rabiee, R. L. Kashyap, and S. R. Safavian, "Multiresolution segmentation-based image coding with hierarchical data structures," in *Proc. IEEE Int. Conf. on Acoustics, Speech and Signal Proc.*, pp. 1870–1873, 1996.

[92] M. Ropert, F. M. de Saint-Martin, and D. Pele, "A new representation of weighted order statistic filters," *Signal Processing*, vol. 54, no. 2, pp. 201–206, 1996.

[93] A. Rosenfeld, ed., *Image Modeling.* New York, NY.: Academic Press, 1981.

[94] M. J. Sabin, "Convergence and consistency of fuzzy c-means/ISODATA algorithms," *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. 9, no. 5, pp. 661–668, 1987.

[95] H.-P. Schwefel, *Numerical Optimization of Computer Models.* Chichester: Wiley, 1981.

[96] H.-P. Schwefel, *Evolution and Optimum Seeking.* New York: Wiley, 1995.

[97] J. Serra, *Image Analysis and Mathematical Morphology.* New York, NY.: Academic Press, 1982.

[98] J. J. Shen and S. S. Castan, "An optimal linear operator for step edge detection," *CVGIP: Graphical Models and Image Processing*, vol. 54, no. 2, pp. 112–133, 1992.

[99] L. Shen and M. M. Rangayyan, "A segmentation-based lossless image coding method for high-resolution medical image compression," *IEEE Trans. Medical Imaging*, vol. 16, no. 3, pp. 301–307, 1997.

[100] V. Srinivasan, "Edge detection using neural networks," *Pattern Recognition*, vol. 27, no. 12, pp. 1653–1662, 1994.

[101] A. Tekalp, H. Kaufman, and J. Woods, "Edge-adaptive Kalman filtering for image restoration with ringing suppression," *IEEE Trans. Acoust., Speech, Signal Processing*, vol. 37, no. 6, pp. 892–899, 1989.

[102] R. F. Thompson, *An Introduction to Neuroscience.* New York, NY.: W.H. Freeman & Company, 1985.

[103] J. T. Tou and R. C. Gonzalez, *Pattern Recognition Principles.* Reading, Massachusetts: Addison-Wesley, 1974.

[104] H. J. Trussell and S. Fogel, "Identification and restoration of spatially variant motion blurs in sequential images," *IEEE Trans. Image Processing*, vol. 1, no. 1, pp. 123–126, 1992.

[105] C. Y. Tyan and P. P. Wang, "Image processing-enhancement, filtering and edge detection using the fuzzy logic approach," in *2nd IEEE Int. Conf. on Fuzzy Systems*, pp. 600–605, 1993.

[106] D. Wang, C. Labit, and J. Ronsin, "Segmentation-based motion-compensated video coding using morphological filters," *IEEE Trans. Circuits and Systems for Video Technology*, vol. 7, no. 3, pp. 549–555, 1997.

[107] L. Wang, S. Z. Der, and N. M. Nasrabadi, "Automatic target recognition using a feature-decomposition and data-decomposition modular neural network," *IEEE Trans. Image Proc.*, vol. 7, no. 8, pp. 1113–1121, 1998.

[108] L. Wang, S. A. Rizvi, and N. M. Nasrabadi, "A modular neural network vector predictor for predictive image coding," *IEEE Trans. Image Proc.*, vol. 7, no. 8, pp. 1198–1217, 1998.

[109] P. H. Winston, *Artificial Intelligence.* Reading, MA.: Addison-Wesley, 1984.

[110] H. S. Wong and L. Guan, "Adaptive regularization in image restoration using a model-based neural network," *Optical Engineering*, vol. 36, no. 12, pp. 3297–3308, 1997.

[111] H. S. Wong and L. Guan, "Adaptive regularization in image restoration using evolutionary programming," in *Proc. IEEE Int. Conf. on Evolutionary Computation*, pp. 159–164, 1998.

[112] J. W. Woods, "Two-dimensional discrete Markovian fields," *IEEE Trans. Inform. Theory*, vol. 18, no. 2, pp. 232–240, 1972.

[113] L. A. Zadeh, "Fuzzy sets," *Information and Control*, vol. 8, no. 3, pp. 338–353, 1965.

[114] L. A. Zadeh, "Fuzzy algorithms," *Information and Control*, vol. 12, no. 2, pp. 94–102, 1968.

[115] L. A. Zadeh, "Fuzzy sets as a basis for a theory of possibility," *Information and Control*, vol. 1, no. 1, pp. 3–28, 1978.

[116] L. A. Zadeh, "Outline of a new approach to the analysis of complex systems and decision processes," *IEEE Trans. Syst. Man Cybern.*, vol. 3, no. 1, pp. 28–44, 73.

[117] P. Zemperoni, "Some adaptive rank order filters for image enhancement," *Pattern Recognition Letters*, vol. 11, no. 2, pp. 81–86, 1990.

[118] Y. T. Zhou, R. Chellappa, A. Vaid, and B. K. Jenkins, "Image restoration using a neural network," *IEEE Trans. Acoust., Speech, Signal Processing*, vol. 36, no. 7, pp. 1141–1151, 1988.