1

MICROCOPY RESOLUTION TEST CHART
NATIONAL BUREAU OF STANDARDS
STANDARD REFERENCE MATERIAL 1010a
(ANSI and ISO TEST CHART No. 2)

National Library
of Canada

Bibliothèque nationale
du Canada

Canadian Theses Service Service des thèses canadiennes

Ottawa, Canada
K1A 0N4

NOTICE

The quality of this microform is heavily dependent upon the quality of the original thesis submitted for microfilming. Every effort has been made to ensure the highest quality of reproduction possible.

If pages are missing, contact the university which granted the degree.

Some pages may have indistinct print especially if the original pages were typed with a poor typewriter ribbon or if the university sent us an inferior photocopy.

Reproduction in full or in part of this microform is governed by the Canadian Copyright Act, R.S.C. 1970, c. C-30, and subsequent amendments.

AVIS

La qualité de cette microforme dépend grandement de la qualité de la thèse soumise au microfilmage. Nous avons tout fait pour assurer une qualité supérieure de reproduction.

S'il manque des pages, veuillez communiquer avec l'université qui a conféré le grade.

La qualité d'impression de certaines pages peut laisser à désirer, surtout si les pages originales ont été dactylographiées à l'aide d'un ruban usé ou si l'université nous a fait parvenir une photocopie de qualité inférieure.

La reproduction, même partielle, de cette microforme est soumise à la Loi canadienne sur le droit d'auteur, SRC 1970, c. C-30, et ses amendements subséquents.

**Kalman Filtering:
Modelling and Restoration of Noisy Images**

by

HOSNI B. H. BELAIFA, B.Sc.

A thesis submitted to
the Faculty of Graduate Studies and Research
in partial fulfilment of
the requirements for the degree of
Master of Engineering

Department of Mechanical and Aerospace Engineering

Carleton University
Ottawa, Ontario
December, 1990

© Copyright
1990, Hosni B. H. Belaifa

National Library
of Canada

Bibliothèque nationale
du Canada

Canadian Theses Service Service des thèses canadiennes

Ottawa, Canada
K1A 0N4

The author has granted an irrevocable non-exclusive licence allowing the National Library of Canada to reproduce, loan, distribute or sell copies of his/her thesis by any means and in any form or format, making this thesis available to interested persons.

The author retains ownership of the copyright in his/her thesis. Neither the thesis nor substantial extracts from it may be printed or otherwise reproduced without his/her permission.

L'auteur a accordé une licence irrévocable et non exclusive permettant à la Bibliothèque nationale du Canada de reproduire, prêter, distribuer ou vendre des copies de sa thèse de quelque manière et sous quelque forme que ce soit pour mettre des exemplaires de cette thèse à la disposition des personnes intéressées.

L'auteur conserve la propriété du droit d'auteur qui protège sa thèse. Ni la thèse ni des extraits substantiels de celle-ci ne doivent être imprimés ou autrement reproduits sans son autorisation.

ISBN 0-315-68889-0

Canada

The undersigned recommend to
The Faculty of Graduate Studies and Research
Acceptance of the thesis

**Kalman Filtering:
Modelling and Restoration of Noisy Images**

submitted by
Hosni B. H. Belaifa, B.Sc.
in partial fulfilment of the requirements for
the degree of Master of Engineering in Mechanical Engineering

A handwritten signature in dark ink, appearing to read 'Hosni Belaifa', is written over a solid horizontal line.

Thesis Supervisor

A handwritten signature in dark ink, appearing to read 'J. M. ...', is written over a solid horizontal line.

Chairman, Department of Mechanical and Aerospace Engineering

Carleton University

Abstract

One of the inherent problems in model based techniques, for the restoration of noisy images, is the blurring and smearing of edges which carry most of the details in the image itself. Kalman filtering is one of the possible estimators used for restoration. However, the artifacts around the edges are still pertinent and the illposed problem of modelling is one of the main sources for the Gaussian noise assumption. A more realistic assumption for the noise would be a mixture of a Gaussian and a small percentage of impulse noise (outliers).

We develop a robust procedure to estimate the model parameters and the image intensity from a degraded image. This model is a modification of the robust procedure proposed by Kashyap and Eom [27], combined with the reduced update Kalman filter developed by Woods and Radewan [52].

We develop an iterative algorithm to restore realistic images, corrupted by impulse noise, using a modified reduced update Kalman filter (RUKF) in conjunction with the robust parameter estimation algorithm. Generally the image field is neither homogeneous nor stationary, therefore the image does not obey a single model. We partition the image into small blocks, say 8×8 , where the image is assumed to obey a nonsymmetric half-plane (NSHP) autoregressive model. The robustness of the iterative reduced update Kalman filter was tested on several images and the performance of the algorithm compared favorably to some commonly used techniques such as the median filter, the RUKF [52], and the robust model [27].

Acknowledgements

This thesis dissertation is the fruit of a great deal of research work. I would like to express my deep gratitude to my thesis supervisor, Dr. H. M. Schwartz, for his guidance in formulating the objectives of this work, and for his encouragement and support throughout the research.

I would also like to thank my fellow student Ian McLaren for his help in familiarizing with the imaging system and the equipment in the image processing laboratory. I extend my appreciations to Dr. Talib Janabi for his criticism and many fruitful discussions and constructive ideas that were brought along. I gratefully appreciate the support and assistance from the Department of Mechanical and Aerospace Engineering.

I would like to express my deep gratitude to my parents and family, especially my wife, to whom I dedicate this thesis for their patience and sacrifice.

Table of Contents

Acceptance Sheet	ii
Abstract	iii
Acknowledgements	iv
Table of Contents	v
List of Figures	viii
List of Tables	ix
1 Introduction	1
1.1 Perspective and Motivation	1
1.2 Image Restoration	4
1.3 Research Overview	10
2 A Sequential Kalman Filter for the Restoration of Noisy Images	13
2.1 Introduction	13
2.2 Image Modelling	14

2.3	Problem Statement	19
2.4	The Sequential Kalman Filter	20
2.4.1	Prediction	20
2.4.2	Update	23
2.5	Initialization of the Algorithm	26
2.6	Local Spatial Statistics Measurement	28
2.7	Simulation and Results	31
3	Robust Modelling for Image Restoration	40
3.1	Introduction	40
3.2	Robust Modelling	42
3.2.1	The Nonsymmetric Half-Plane Autoregressive Model	42
3.2.2	The Cost Function	45
3.3	The Minimization Algorithm	48
3.4	Parameter Estimation with Noisy Image and Image Restoration	51
3.5	Simulation	55
4	A Modified Reduced Update Kalman Filter for Image Restoration: Robust Modelling	62
4.1	Introduction	62
4.2	The Reduced Update Kalman Filter	64
4.3	New Development	71
4.4	Simulation and Results	81
5	Summary and Conclusions	93
	Bibliography	98

Appendices	103
Appendix A Definitions	103
Appendix B Listings of Simulation Programs	106

List of Figures

2.1	The nonsymmetric half plane (NSHP)	16
2.2	The subregion of the NSHP for image modelling	16
2.3	The square-spiral scanning operation	18
2.4	The notation of the causal field	18
2.5	The closest pixels for the state equation	18
2.6	An edge across the window region	30
2.7	The window divided into subregions R_{ij}	30
2.8	The local means for the subregions in the window and the edge orientation map	30
2.9	Images of the girl	34
2.10	The vegetables image	36
2.11	The restored images	38
3.1	The nonsymmetric half-plane	43
3.2	Division of the image into small windows	55
3.3	The girl and baboon pictures	58
3.4	The vegas and square pattern images	60
4.1	The nonsymmetric half-plane for modelling	65
4.2	The partitioning of the state vector	67
4.3	The original images and their noisy versions	85
4.4	Restored images of the girl by different algorithms	87
4.5	Restored images of the baboon by different algorithms	89
4.7	Error measurement along line (30,j)	91

List of Tables

4.1	The mean square-error between the original and the restored images	83
-----	--	----

Chapter 1

INTRODUCTION

1.1 Perspective and Motivation

The sensing capabilities and communication skills acquired by humans are certainly the most valuable tools which enables them to interact effectively with the environment. Besides speech, images provided by the human visual system are a primary medium of communication and interpretation. A single image embodies an enormous amount of information that cannot be carried, with the same precision and flexibility, by any other sensory means. With the advances in digital computing and the variety of image processing techniques developed so far, we are able to obtain digitized images corresponding, to a large extent, to the natural visible information.

Digital pictures are formed by means of imaging systems that resemble the human visual system. The purpose of these systems is to make available information that cannot be directly perceived by sight. These imaging systems, whether they

function by means of optics, photography, television, X-ray, etc., consist of three types of devices: a video signal sensor, a device for transforming and transmitting the video signal, and a picture synthesizer. The sensor interacts directly with the observed object. The picture synthesizer forms the image, making it accessible for direct visual perception. The video signal transforming and transmitting device matches and links the sensor and the picture synthesizer.

Techniques for digitizing images vary as much as the purposes of using these images. However, the common activities that tie the several applications are the capture, storage, and interpretation of the information embodied in an image, and the necessity to improve images that are not of optimum utility for the purpose at hand. The quality of an image is critical in many applications. For example, using video images as the guiding media for industrial robots to detect and pick up objects on conveyor belts may sometimes lead to disastrous operations if the images are degraded (e.g. "snow flakes", blurring, and distortion).

Among the images obtained everyday, some are of lesser quality, which could be of such importance or are so unique, that it is necessary to consider the techniques by which the image quality may be enhanced or the techniques by which the degrading phenomena may be removed and the undegraded image restored. In medicine, for instance, X-ray images play a capital role in the diagnostics and interpretation of body illnesses and other biomedical applications. In archaeology, there have been situations where a valuable work of art has been damaged and only a degraded photography was available where some of these techniques were used successfully to restore such images. Several other applications in many scientific disciplines can be found in geography, astronomy, biology, law enforcement, industrial applications just to mention a few.

Over the past three decades, such problems have motivated the development of many techniques for the restoration and noise reduction in images. One early fruitful application of image processing was conducted at the Jet Propulsion Laboratory of the California Institute of Technology in the early 1960's as part of the Apollo program. Since then, the development of better, more efficient and cheaper

image encoders, analyzers, and display devices has made spatial information more readily available in many different digital forms, for a variety of areas. However, the development of large-scale digital computers and the advances in space programs led researchers to establish several techniques to improve the visual quality of the images received from the Ranger 7 space probe. These techniques served as the basis for improved methods used in the enhancement and restoration of images from such familiar programs as the Apollo manned flights to the moon.

Even with these advances in imaging systems, digital images taken in realistic environments, inevitably suffer from various types of degradations including noise, blurring, geometric distortions etc.. An image may be effected by noise from a wide variety of sources, including sensor noise, grain noise, and channel noise. The noise may be systematic or random, the former being the easier to deal with in practice. The presence of such degradations often leads to poor quality images and the loss of valuable details, such as edges, is inevitable.

The application of image processing techniques may vary from one area to another, but the basic mathematical principles of filtering, enhancement, and restoration are the same. The material contained in this dissertation is mainly focused on the restoration of digital images corrupted by additive noise. Modelling the image and the degradation phenomena is the most commonly used technique for image processing applications. In the chapters to follow, three principal aspects of the problem are presented:

1. Development of mathematical models for the test images used;
2. Development of algorithms capable of restoring images in noisy environments given explicit criteria;
3. Computer simulations to verify the proposed algorithms.

The following section is a literature review about closely related material to the restoration and enhancement of realistic images. This survey is an overview of the most commonly used techniques for the restoration of images corrupted by noise or both noise and blur.

1.2 Image Restoration

Images acquired through recording media are often corrupted by various forms of degradations among which noise and blur are the most prominent. The quality of a received image is limited by the receiver and the environment within which the image is taken. The image sensor is also a major factor. As the image sensor operates at its sensitivity limit, the noise associated will inevitably manifest itself in the received image, such as grain noise and shot noise (Andrews and Hunt, [6]). Blurring is often caused by the relative motion of objects and imaging systems, atmospheric turbulence, out of focus operation, physical limits of image sampling systems etc..

Corrupted images often pose problems for analysis and detection. Techniques for improving the visual quality of noisy images and recovering images from their degraded counterparts are referred to as image enhancement and restoration. Earlier research in this area was to develop filters in the frequency domain (Andrews, [5]; Andrews and Hunt, [6]). The restoration process becomes a deconvolution or an inverse problem. The use of mathematical transformations such as the fast Fourier transform (FFT) resulted in considerable gains in terms of computation. However, due to the limitations of these approaches, the images were assumed to be linear and stationary processes, and the noise degradations were considered as broad band (white) or band limited which is not the case for most images.

From the basic mathematical principles of linear filtering and prediction [11],[44], a variety of estimation (restoration) algorithms have been introduced since the early 1960's. These techniques could be classified into three large classes of

estimators: an M-estimator which is a maximum likelihood-type estimator, and is obtained by solving a minimization problem; an L-estimator which is a combination of ordered statistics; and an R-estimator that is derived from the rank tests.

Early attempts to extend the one-dimensional Bayesian estimation technique (Habibi [17]) led to the development of stochastic restoration algorithms in the spatial domain. In these algorithms, the images are assumed to be random processes (usually stationary). As will be seen, these techniques are, for the most part, variations of two-dimensional Kalman-type filters which fall into the maximum likelihood category of estimators since they are solving a minimization problem, i.e. minimize the post update square error.

Kalman filter theory has long been applied to one-dimensional systems with great success due to its high efficiency. This processor is a filter of the recursive type. It was developed as a generalization of the nonrecursive Wiener filter. Many researchers attempted to extend this powerful estimator to two-dimensional applications which could be perfectly fitted to the image restoration problem. The first attempts for two-dimensional image restoration were by Nahi and Assefi [37] who introduced the Bayesian estimation technique for recovering images corrupted by additive white Gaussian noise. They introduced the Kalman procedure to recursively determine the minimum mean-square error estimate of the image. Further to this, Nahi and Habibi [38] extended this technique to restore images corrupted by both noise and blur.

In order to implement the Kalman filter in the spatial domain, we have to establish an artificial causal image model which contains image vectors of very large dimension. Due to the fact that any image is of finite extent we are unable to establish a global state space model as we do for the time-domain Kalman filter. The early attempts to extend Kalman filtering to two-dimensions, to achieve a truly recursive processor, were of limited success. This is mainly due to both the difficulties in establishing a suitable two-dimensional recursive model and also to the high dimensionality of the resulting state vector.

With a view to overcome the high dimensionality and computational problems, Woods and his colleagues [47]-[52]-[53]-[54] proposed a special structure for image state vectors which takes only the closest pixels before and above the pixel being estimated. Based on these image vectors, they were able to establish a local state-space representation required by the Kalman filter. Woods and Radewan [52] proposed two schemes which to a large extent overcome the computational problems that have precluded the use of two-dimensional Kalman-type processors. They developed two new approximations: one to the 2-D Kalman vector processor, which updates a line at a time, called the strip Kalman filter; and another to the 2-D scalar processor, which updates one point at a time, called the Reduced Update Kalman Filter (RUKF). The latter processor is shown to be optimal in the sense that it minimizes the post update mean square error. The amount of computations is considerably reduced since only those pixels in the immediate vicinity are used for modelling. Similarly, Murphy and Silverman [35], considered two other suboptimal restoration schemes, namely strip restoration and constrained optimal restoration. In the latter scheme, the gain matrix of the Kalman filter is constrained to be square.

The advantages of two-dimensional recursive filters are that they require less computation time than the nonrecursive ones. They handle space-variant blur easily, and, as is well known (Anderson and Moore [3]) being Kalman-type filters, they are capable of coping with the nonstationarity problems.

Traditionally, for the ease of computation and mathematical derivation, an image is normally assumed to be a wide-sense stationary field. Therefore, the statistical properties of the image are governed globally rather than locally by its stationary covariance matrix which is of Toeplitz form [6]. Under this assumption, the necessary computations can be carried out by FFT algorithms and consequently, the computation time is reduced drastically. The savings in computation time are, however, at the expense of restoration quality. Algorithms based on this assumption tend to globally smooth out the noise and any abrupt changes of image intensities as well. As a result, the restored images are smooth but blurred. For example, Biemond and his colleagues [7] derived a fast Kalman filter for the nearly optimal recursive

restoration of images degraded in a deterministic way by blur and in a stochastic way by additive white Gaussian noise. A parallel set of N dynamical models suitable for the derivation of N low order vector Kalman filters, in the frequency domain, is obtained. They approximated the band-Toeplitz structure of the model matrices and the degradation matrices by circulant matrices. Using a semicausal model along the properties of circulant matrices, the diagonalization process of these matrices was accomplished by means of the FFT, which resulted in considerable gains in terms of computation.

Recent studies in human vision have shown that edges are critical for pattern recognition and the perceptual registration of images. This observation suggests that the abrupt changes in image intensities should be retained as much as possible, after the restoration, in order to obtain good visual quality. It is intuitively obvious that the visual quality of the restored image can be improved if edges can be preserved. Wallis [50] suggested an algorithm based on local mean and variance. In his approach each pixel is required to have the local average mean and variance, and sudden fluctuation is measured. The algorithm is able to preserve edges in restoration. The main drawback of this technique is that it tends to over-enhance subtle details of the image at the expense of its principle features. Based on the modification of Wallis' model, Lee [31] [32] developed adaptive algorithms with striking results. These techniques, though they lack the mathematical elegance and sophistication of other techniques, are simple and effective. Chan and Lim [10] proposed an interesting and rather simple four-stage one-dimensional adaptive algorithm. In their approach they process the image in cascade along four directions (0, 45, 90, and 135 degrees). The basic idea is that if the one-dimensional filter is oriented in the same direction as the edge, the noise is filtered along the edge, and in doing so, the edges are preserved. The simulation results seem much better than those obtained by its two-dimensional counterpart. Other works on preserving edges can be found in the literature, e.g. Ingle et al. [22], Rajala et al. [42], Abramatic et al. [1], Woods et al. [47], to mention a few.

Research on developing filters for nonstationary, and signal dependent noise processes has recently been published by some authors. After investigating some statistical characteristics of image fields Hunt et al. [21] [48] showed that images can be decomposed into a nonstationary mean Gaussian model and stationary fluctuations about the mean. Ingle and his colleagues [22] used a composite image model that assumes that the image is composed of many different stationary components, and each has a distinct correlation structure. They applied the reduced update Kalman filter (RUKF) to image restoration by establishing a bank of RUKF's running in parallel. They used an identification estimation scheme in which each point is assigned a stationary image model and filtered by the specific Kalman filter. Kuan et al. [30] considered an image model with nonstationary mean and nonstationary variance (NMNV), and developed a recursive algorithm for image restoration. They leave the nonstationarity to the input process, so that the filter has a simple, space-invariant structure. In their scheme, it becomes unnecessary to construct a space-variant dynamic model for each image, and only the local mean and the local variance of the original image need to be estimated.

It is known (Netravali et al. [39],[4]; Rajala et al. [42]) that the human visual system is insensitive to noise in the high contrast regions of an image and relatively more sensitive to noise in flat regions. Such perceptual criteria for the development of image models and restoration algorithms can often be effective. For example, Anderson and Netravali [4] defined a subjective error criterion based on the human visual system model and established an adaptive strategy that strikes a compromise between the loss of resolution and noise rejection such that the same amount of noise is suppressed throughout the entire image. Abramatic and Silverman [1] generalized this procedure in the framework of the classical Wiener filter. Motivated by the earlier work Rajala and DeFigueiredo [42] proposed a recursive least-squares method for the restoration of an image distorted by motion blur and corrupted by additive white Gaussian noise. They combined the so-called visibility function [4] with the Kalman filter algorithm. The procedure was conducted by segmenting the image into disjoint regions according to local spatial activity of the region, and then determined

the covariance structures of these segments. In this framework, the filter adapts itself to each segment through the visibility function for nonstationary restoration.

An interesting restoration method was proposed by Zhe Wu [55] who used a multidimensional Kalman filtering approach to restore images degraded by both noise and blur. The three-dimensional models extend the regions of the correlation of the picture elements and of the point spread function (PSF) of blur to a nonsymmetric half-plane. Furthermore, by merging these two models, the image process and the PSF of the blur, he came up with a single model that reduces the dimension of the state vector, and consequently significant reductions in the amount of computations and storage requirements resulted. A strip Kalman processor, scanning two lines at a time, was presented and tested. The simulation results show that this 3-D model works effectively. Strickland [46] described a method for estimating the correlation parameters of first-order Markov (nonseparable exponential) autocovariance models. The method assumes that image data are stationary within $N \times N$ pixel sub-blocks. A clear improvement of the filter performance and substantial gains in terms of computation time and storage were obtained. However, with the late improvements to the reduced update Kalman filter, it was possible to develop an edge-adaptive Kalman processor for image restoration. Tekalp, Kaufman and Woods [47] extended the two-dimensional linear space-invariant (LSI) reduced update Kalman filter to an edge adaptive space-variant filter for the restoration of noisy and blurred images using a decision-directed approach. The edge-adaptive reduced update Kalman filter was motivated by the need to suppress the ringing artifacts caused by the LSI processing. They showed that the ringing artifacts can be suppressed, to a great extent, by using multiple image models and a maximum a posteriori decision procedure that provide a better match to the local edge orientation.

For reasons of mathematical and computational convenience, the assumptions upon which the theoretical framework is established are often simplified and idealized. This sometimes leads to results that are far from being realistic. In the development of numerous image restoration algorithms, the image is considered as a multivariate Gaussian distribution. The noise is usually assumed to be additive,

white, and Gaussian. For instance, the popular Wiener filter is simplified by this stationary assumption, while the Kalman filter requires that the system be linear, finite dimensional, and driven by white noise (Sage and Melsa [44], Deutsch [11]). In contrast to these, many physical noise processes do not possess such nice properties. For instance, the image recorded by a vidicon sensor is usually a highly structured entity and does not possess stationary statistics [16]. Further to this, the noise from a vidicon sensor is coupled into the signal by the shot-noise process of receiving individual photons. Such noise is signal-dependent and nonstationary. Many other physical noise processes such as Poisson noise and multiplicative noise are inherently signal-dependent. Blurring effect is generally space-variant. Consequently, the simplified assumptions which are often required by the existing filtering techniques may cause difficulty in practice.

For instance, the Gaussian assumption is not appropriate for parameter estimation in images which are degraded by impulse noise, often called salt and pepper noise. The conventional least-squares algorithm is not appropriate for parameter estimation since the presence of outliers in the data can cause a catastrophe. However, robust modelling are the most appropriate techniques that could be used in such cases. Kashyap and Eom [27] proposed a robust algorithm to estimate the NSHP autoregressive model parameters and the original image intensity simultaneously from the impulse noise corrupted image, where the model governing the image is not available. The robustness of the algorithm was proven by simulation and the proposed restoration procedure was tested and compared favourably with some traditional methods such as the median filter or the α -trimmed mean filter. This model also performed very well around edges (no artifacts).

1.3 Overview of the Research

The main course of this thesis research is to develop some techniques to improve the visual quality of realistic images. Motivated by the fact that the Kalman

filter can be applied to the restoration of images, some new approaches have been developed to reduce the effect of different types of additive noise which may or may not satisfy the Gaussian assumption.

As we discussed earlier, most of the stochastic restoration techniques are variations of the Kalman-type filters. This shows the effectiveness of the Kalman filter as a restoration processor. Motivated by this fact, an adaptive sequential Kalman filter was developed (chapter 2). A detailed formulation has been directed with a precise mathematical development of the Kalman equations. This sequential estimator treats the two-dimensional image data array as a long string of pixel intensities. The resulting scalar Kalman processor behaves like a one-dimensional Kalman estimator, which has been successfully applied. Raster scanning is the most commonly used operation for image processing tasks. Since the sequential Kalman filter is of the recursive type, a spiral scanning operation made it possible to treat the image as a one-dimensional array. The image data is processed one pixel at a time. The procedure was tested on several images corrupted by a white Gaussian noise of different intensities. A qualitative and quantitative evaluation for the efficiency of the algorithm was conducted.

In most model-based restoration techniques, the noise is assumed to be a multivariate Gaussian distribution. However, this is not the case for many realistic images. Chapters 3 and 4 consist of restoration methods that treat the noise somewhat differently. The observation noise was assumed to be a contaminated white Gaussian noise. It is a more realistic assumption in which the noise is a mixture of a Gaussian and outliers.

In chapter 3, a robust model was developed for the parameter estimation in a noisy environment, and a restoration procedure is conducted for the recovery of realistic images corrupted by impulse noise. This technique is an iterative restoration scheme which models the image and the noise effectively.

Chapter 4 is new development in image restoration. A new algorithm is presented, in which both Kalman filter theory and robust modelling technique are combined to restore images corrupted by a mixture of a white Gaussian noise and an

impulse noise, commonly called outliers. The resulting algorithm performs well and compares favourably to some of the commonly used methods.

Chapter 2

A SEQUENTIAL KALMAN FILTER FOR THE RESTORATION OF NOISY IMAGES

2.1 Introduction

Kalman filtering has been applied successfully to one-dimensional systems. The extension of this processor to two-dimensional systems (i.e. image processing) is difficult but promising. The research in the field goes back to the early 1970's, when Nahi and Habibi [38] developed a model for two-dimensional images and used the recursive Kalman processor to estimate images from their noisy versions. Further development of 2-D Kalman filters has been carried out by many others such as

Powell and Silverman [41], Woods et al. [22][28][47][51], Murphy and Silverman [35], Kondo et al. [29], Wu [55], just to mention a few.

Conventionally, it is assumed that the image field is wide-sense-stationary. The statistical properties of the image field are globally characterized by the stationary covariance function of the image. Unfortunately, in most cases, the image field does not have stationary statistics, and restoration algorithms based on the stationarity assumption are insensitive to abrupt changes in image intensity (i.e. edges). Consequently, these algorithms tend to smooth out the edges, and the restored image is usually blurred. More recently, many researchers have proposed different image models and algorithms to handle the nonstationarity of the image statistics. Local stationarity rather than the global stationarity of the image has been proposed for most of these algorithms.

In the present chapter, a sequential Kalman filter is developed. The Kalman processor is a scalar estimator in which a single pixel is processed at a time. The two-dimensional image data array is considered as a long string of pixels which behaves as a one-dimensional system. However, in order to obtain better visual quality for the restored images, the image field is assumed to be nonstationary. The sequential Kalman filter is based on a causal state-space image model. This algorithm includes some local statistics measurements in the filter gain such that the restored image would retain the edge information.

2.2 Image Modelling

It is well known that, for most images, significant correlations exist only between pixels close to each other. Therefore, the image can be modeled over a finite area around the pixel under consideration (Rosenfeld and Kak, [43]; Woods and Radewan, [52]).

For computational and dimensionality problems, the image is modeled over a nonsymmetric half plane (NSHP). This is the case for all the model based techniques for image restoration applications. The nonsymmetric half plane (NSHP) is described in Figure 2.1. The pixels which have been processed are called "Past", the pixel presently processed is called "Present", and those pixels to be processed are called "Future". These definitions are rather artificial since the scanning operation can take any order. If (m,n) is the present coordinate position, $P(m,n)$ is the "Past" region which is defined as,

$$P(m,n) = \{(i,j) : [1 \leq i \leq M; 1 \leq j \leq n-1] \cup [1 \leq i \leq m-1; j=n]\} \quad (2.1)$$

where M is the width of the image field, m is the horizontal coordinate, and n is the vertical coordinate of the image field. The image signal can be modeled by a nonsymmetric half plane autoregressive model as

$$f(m,n) = \sum_{(i,j) \in P(m,n)} a(i,j) f(m-i, n-j) + w_s(m,n) \quad (2.2)$$

where $w_s(m,n)$ is the modelling error which is assumed to be a white Gaussian noise with zero mean and variance σ_{ws}^2 . This assumption can be used only for autoregressive models, otherwise $w_s(m,n)$ is not white [37]. The parameters $a(i,j)$ are the modelling coefficients. Let $g(m,n)$ be the observed image which is the image signal $f(m,n)$ corrupted by an additive Gaussian noise $v(m,n)$ where,

$$g(m,n) = f(m,n) + v(m,n). \quad (2.3)$$

The noise $v(m,n)$ is assumed to be a white Gaussian noise with zero mean and variance $\sigma_v^2(m,n)$, and is independent of both the image $f(m,n)$ and the driving noise

Figure 2.1- The Nonsymmetric Half Plane (NSHP).

Figure 2.2 The subregion of the NSHP for image modelling.

$w_s(m,n)$. The model in equations (2.2) and (2.3) will lead to high dimensionality problems. The excessive computations involved are due to the high dimensionality of images (i.e. 512x512 images). Therefore, since only those picture elements closest to the present pixel are significant to the modelling of the image intensity $f(m,n)$, where significant correlations exist, equation (2.2) can be reduced to:

$$f(m, n) = \sum_{(i,j) \in \mathcal{R}} a(i, j) f(m-i, n-j) + w_s(m, n), \quad (2.4)$$

where $\mathcal{R} = \{(k,l)\}$ is a subregion of the NSHP $P(m,n)$, Figure 2.2. Without loss of generality, assume that the global mean of the image is zero, that is

$$E[f(m,n)] = 0. \quad (2.5)$$

The number of computations can be reduced by considering the image data as a long string of pixels and the Kalman processor will be a one-dimensional estimator. This can be achieved by a spiral scanning operation. The spiral scanner runs continuously along square-shaped spiral lines (Figure. 2.3), and ends at the geometrical center of the image. For this particular spiral scanning operation, the image model (2.4) can be rewritten for the subregion Ω (Figure 2.4) of the nonsymmetric half plane as

$$f(m, n) = \sum_{(i,j) \in \Omega} a(i, j) f(m-i, n-j) + w_s(m, n). \quad (2.6)$$

Now using the orthogonality principle [40], which states that the linear minimum variance estimation (LMV) $\hat{y} = Ax + B$ of y is such that the estimation

Figure 2.3 The Square-Spiral scanning operation.

Figure 2.4 The notation of the causal field.

Figure 2.5 The closest pixels for the state equation.

error defined as ($e = y - \hat{y}$) is orthogonal to the data x , or $\hat{y}$ is the orthogonal projection of y onto X , where $X = \{x\}$. This can be translated into,

$$E[(y - \hat{y}).x] = 0. \quad (2.7)$$

In the present development, we require an estimate $\hat{f}$ of f given the observation g such that $\hat{f}$ is the perpendicular projection of f onto P , or $\hat{f} = \hat{E} [f/P]$. Here we have used $\hat{E}$ rather than E to indicate that it is not the true conditional mean, in the sense that the linear minimum variance estimate is a weighted linear combination of the observations.

2.3 Problem Statement

Given the models in equations (2.2) and (2.3), determine an estimate $\hat{f}(m,n)$ of $f(m,n)$, which is a linear combination of the observation $g(m,n)$ and estimates $\{\hat{f}(i,j) \in \Omega\}$. The estimate is optimal subject to the criterion that the expected value of the sum of the variance between f and $\hat{f}$ is minimum, that is $\hat{f}(m,n)$ is to be chosen such that,

$$Q(m,n) = E [(f(m,n) - \hat{f}(m,n))^2] = \text{minimum}. \quad (2.8)$$

We may write the estimate $\hat{f}(m,n)$ as

$$\hat{f}(m,n) = \hat{f}^-(m,n) + K(m,n) e(m,n) \quad (2.9)$$

where $K(m,n)$ is the Kalman gain which has to be computed, $e(m,n)$ is the residual, and $\hat{f}^-(m,n)$ is the predicted estimate of the original image.

2.4 The Sequential Kalman Filter

The Kalman processor is an estimator which is composed of two steps: the prediction step; and the update step. The prediction part consists of predicting the image signal based on the image model, and determining the a priori error covariance (covariance of the prediction error). The second part will calculate the Kalman gain and updates of both the image estimate and the error covariance. In this section we will derive the Kalman equations based on the assumptions made earlier.

2.4.1 Prediction

The predicted estimate of the image, at location (m,n) can be described as

$$\hat{f}(m, n) = \sum_{\mathbf{a}} a(i, j) \hat{f}(m-i, n-j) . \quad (2.10)$$

The innovation process is obtained by:

$$e(m,n) = g(m,n) - \hat{f}(m,n), \quad (2.11)$$

$e(m,n)$ is the residual, which plays an important role in the approaches to adaptive filtering (Ljung and Söderström, [33]). It is easy to show that if the minimum variance estimator is unbiased, that is $E[\hat{f}(m,n)] = f(m,n)$, the innovation process is also white with zero mean. Equation (2.11) can be written as follows,

$$e(m,n) = \tilde{f}(m,n) + v(m,n) \quad (2.12)$$

where,

$$\tilde{f}(m,n) = f(m,n) - \hat{f}(m,n). \quad (2.13)$$

Using the orthogonality principle [40] and equations (2.8) and (2.12) we have,

$$\begin{aligned} E[e(m,n)e(k,l)] &= E\{\{\tilde{f}(m,n) + v(m,n)\}\{g(k,l) - \hat{f}(k,l)\}\} \\ &= E[\tilde{f}(m,n)g(k,l)] + E[\tilde{f}(m,n)\hat{f}(k,l)] + \\ &\quad E[v(m,n)g(k,l)] + E[v(m,n)v(k,l)]. \end{aligned} \quad (2.15)$$

In the above equation, the prediction error $\tilde{f}(\cdot)$ is equal to the system noise w_s . Since the system noise is independent of the data and the observation noise, the right hand side of equation (2.15) is equal to zero for $m \neq k$ and $n \neq l$. Therefore we have

$$E[e(m,n) e(k,l)] = E[v(m,n) v(k,l)] = 0 \quad \text{for } m \neq k; n \neq l \quad (2.16)$$

and obviously, since $w_s(m,n)$ is Gaussian and white we have

$$E[w_s(m,n)] = 0, \quad (2.17)$$

with the innovation process being defined as

$$e(m,n) = \tilde{f}(m,n) + v(m,n). \quad (2.18)$$

From equation (2.18) the expected value of $e(m,n)$ is

$$E[e(m,n)] = E[\tilde{f}(m,n)] + E[v(m,n)], \quad (2.19)$$

therefore we obtain

$$E[e(m,n)] = 0 \quad (2.20)$$

which implies that the innovation $e(m,n)$ is a white process. The covariance of the innovation process can be written in the general form as

$$\begin{aligned} E[e(m,n)e(k,l)] &= (V_{\tilde{f}}(m,n;k,l) + \sigma_v^2(m,n))\delta_{mk}\delta_{nl} \\ &= R_e(m,n)\delta_{mk}\delta_{nl} \end{aligned} \quad (2.21)$$

where,

$$R_e(m,n) = [V_{\tilde{f}}(m,n; m,n) + \sigma_v^2(m,n)], \quad (2.22)$$

and,

$$V_{\tilde{f}}(m,n; k,l) = E[\tilde{f}(m,n) \tilde{f}(k,l)], \quad (2.23)$$

which is the a priori error variance, and δ is the kronecker delta function defined as,

$$\delta = \begin{cases} 1, & r = s \\ 0, & r \neq s \end{cases} \quad (2.24)$$

In the following we shall express the a priori variance as a function of the estimation error $\tilde{f}(m,n)$. Knowing that $\tilde{f}(m,n) = f(m,n) - \hat{f}(m,n)$, from the image model (2.6) we have:

$$f(m,n) = \sum_{(i,j) \in \Omega} a(i,j) f(m-i, n-j) + w_s(m,n), \quad (2.25)$$

where $w_s(m,n)$ is a white random process with variance σ_w^2 . Thus equation (2.23) can be written as

$$\begin{aligned}
V_{\tilde{f}}(m, n; m, n) &= E[\tilde{f}(m, n)\tilde{f}(m, n)] \\
&= E\left[\sum_{\Omega} \sum_{\Omega} a(i, j)\tilde{f}(m-i, n-j)\tilde{f}(m-k, n-l)a(k, l)\right] + \sigma_{ws}^2(m, n) \\
&= \sum_{\Omega} \sum_{\Omega} a(i, j)E[\tilde{f}(m-i, n-j)\tilde{f}(m-k, n-l)]a(k, l) + \sigma_{ws}^2(m, n) \\
&= \sum_{\Omega} \sum_{\Omega} a(i, j)V_{\tilde{f}}(m-i, n-j; m-k, n-l)a(k, l) + \sigma_{ws}^2(m, n),
\end{aligned}$$

hence the a priori covariance equation is given by

$$V_{\tilde{f}}(m, n; m, n) = \sum_{\Omega} \sum_{\Omega} a(i, j)V_{\tilde{f}}(r, s; t, u)a(k, l) + \sigma_{ws}^2(m, n) \quad (2.26)$$

where $r = m-i$, $s = n-j$, $t = m-k$ and $u = n-l$, respectively. $V_{\tilde{f}}(r, s; t, u)$ is the a posteriori variance which is readily available.

2.4.2 - Update

This part consists of determining the Kalman gain and consequently updating the image estimate and the a posteriori covariance at location (m, n) .

Gain equation

From equation (2.7) we have:

$$E[\{f(m,n) - \hat{f}(m,n)\} e(k,l)] = 0, \quad (2.27)$$

which could be written as,

$$\begin{aligned} E[\{f(m,n) - \hat{f}(m,n)\} e(k,l)] &= E[\{f(m,n) - \hat{f}(m,n) - K(m,n)e(m,n)\} e(k,l)] \\ &= E[\{\tilde{f}(m,n) - K(m,n) e(m,n)\} e(k,l)] = 0. \end{aligned} \quad (2.28)$$

By expanding equation (2.28) we obtain,

$$E[\tilde{f}(m,n) e(k,l)] = E[K(m,n)e(m,n) e(k,l)] \quad (2.29)$$

which yields

$$E[\tilde{f}(m,n) e(k,l)] = K(m,n) R_e(m,n) \delta_{mk} \delta_{nl}, \quad (2.30)$$

hence, for the case when $m=k$, $n=l$ we have:

$$K(m,n) = E[\tilde{f}(m,n) e(m,n)] R_e^{-1}(m,n) \quad (2.31)$$

and $K(m,n) = 0$ otherwise. Substituting for $e(m,n)$ into equation (2.31) we get

$$K(m,n) = E[\tilde{f}(m,n) (\tilde{f}(m,n) + v(m,n))] R_e^{-1}(m,n) \quad (2.32)$$

expanding equation (2.32) gives

$$K(m,n) = \{E[\tilde{f}(m,n)\tilde{f}(m,n)] + E[\tilde{f}(m,n)v(m,n)]\}R_e^{-1}(m,n) \quad (2.33)$$

and since the noise is white we have:

$$E[\tilde{f}(m,n) v(m,n)] = 0. \quad (2.34)$$

From equations (2.33) and (2.34), the Kalman gain can now be written as

$$K(m,n) = V_f(m,n; m,n) R_e^{-1}(m,n). \quad (2.35)$$

Given the gain equation (2.35), we can update both the image estimate and the error covariance. The image estimate was described in equation (2.9). Therefore, the a posteriori covariance equation can be derived from the estimation error $\tilde{f}(m,n)$ defined by:

$$\tilde{f}(m,n) = f(m,n) - (\hat{f}(m,n) + K(m,n)e(m,n)) \quad (2.36a)$$

which could be reduced to

$$\tilde{f}(m,n) = \bar{f}(m,n) - K(m,n)e(m,n). \quad (2.36b)$$

The covariance of the post update estimation error is defined as

$$E[\tilde{f}^2(m,n)] = E[(\bar{f}(m,n) - K(m,n) e(m,n))^2] \quad (2.37)$$

which leads to

$$\begin{aligned} V_f(m,n; m,n) = E[\{\bar{f}(m,n)\}^2] - 2K(m,n) E[\bar{f}(m,n)e(m,n)] \\ + K^2(m,n) E[\{e(m,n)\}^2]. \end{aligned} \quad (2.38)$$

Thus,

$$\begin{aligned} V_i(m,n; m,n) = & V_i(m,n; m,n) - \\ & 2K(m,n) E[\{\tilde{f}(m,n)\}^2] + E[\tilde{f}(m,n)v(m,n)] \\ & + K(m,n) V_i(m,n; m,n) \end{aligned} \quad (2.39)$$

by expanding equation (2.39) we obtain

$$\begin{aligned} V_i(m,n; m,n) = & V_i(m,n; m,n) - 2K(m,n) V_i(m,n; m,n) \\ & + K(m,n) V_i(m,n; m,n) \end{aligned} \quad (2.40)$$

this is the updated estimate of the error covariance which could be written in the compact form

$$V_i(m,n; m,n) = [1 - K(m,n)] V_i(m,n; m,n) \quad (2.41)$$

Equations (2.9), (2.10), (2.26), (2.31), and (2.41) are the sequential Kalman filter for the image model (2.2) and the observation (2.3).

2.5 Initialization of the Algorithm

In order to implement the sequential Kalman filter, which is a scalar Kalman filter processing one pixel at a time, we need to set the initial conditions. The image field is a finite two-dimensional array, therefore, the initial conditions for the estimation algorithm are the signal estimates over the image boundaries.

The current estimate of the Kalman filter is optimal as long as the previous estimates are optimal. Further to this, the estimates of the Kalman filter are independent of their initial values, the normal estimates (orthogonal projection of the

data) will asymptotically approach the optimal values even if the initial conditions are not known exactly (Anderson and Moore [3]). Due to the particular modeling method involved, the sequential Kalman filter is suboptimal. Hence, it is necessary to start with proper initial values to initialize the computations.

Several techniques can be used for this purpose. One of these techniques is to carry out the initial restoration over the image boundaries using the following algorithm.

$$\hat{f}(m,n) = f_m + K_b [g(m,n) - f_m] \quad (2.42)$$

where K_b is the gain, f_m is the mean of the original image, and $g(m,n)$ is the corrupted image.

$$K_b = \frac{\sigma_f^2}{\sigma_f^2 + \sigma_v^2}, \quad (2.43)$$

the variance of the image can be calculated by the following approximations, where

$$\sigma_g^2 = (1/A) \sum [g(r) - g_m]^2; \quad g_m = E[g(r)];$$

$$\text{and} \quad f_m = g_m - E[v]; \quad \sigma_f^2 = \sigma_g^2 - \sigma_v^2. \quad (2.44)$$

- A is the dimension of the image field (a window in the image, or the entire image).
- $r = (i,j)$ is the spatial coordinates of pixels in the image.
- g_m , σ_g^2 , σ_f^2 , and σ_v^2 are the mean and variance of the corrupted image, the variance of the original image, and the variance of the observation noise respectively.

Equation (2.42) is basically a linear least squares estimate of the image with the gain K_b as the variance ratio. This equation is itself a simple pixelwise restoration algorithm in the early development of restoration techniques.

2.7 Local Spacial Statistics Measurements

It is well known that edges carry the essential information revealing the details of the image. A simple and effective scheme proposed by Wallis [50] is applied to measure the spatial activity, and the Kalman filter developed is modified to adapt to the local statistics of the image to be processed.

The noise statistics (i.e. σ_v^2) used in this Kalman filter algorithm play an active role in determining the gain factor for the filter. If the variance $\sigma_v^2(m,n)$ is given as a constant throughout the entire image field, the noise in the image is suppressed uniformly leading often to a smooth but blurred restored image. The variance σ_v^2 can be weighted by the local statistics such that the restoration filter will be able to preserve edges to some extent (limit the blurring). Let us define the local statistics as such:

$$\sigma_t^2(m,n) = \sigma_g^2(m,n) - \sigma_v^2(m,n), \quad (2.45)$$

where,

$$\sigma_g^2(m,n) = (1/N_D^2) \sum [g(m,n) - g_m]^2 \quad (2.46)$$

The local statistics are calculated over a window around the present pixel. The parameter g_m is the mean of the observed image, and N_D^2 is the window size.

To adapt to the spatial activity, the noise intensity is weighted by a weighting factor $w_t(m,n)$. The modified sequential Kalman filter will act normally in flat regions suppressing the noise. However, in a region of high spatial activity, the Kalman filter is modified to work on the noise, say $v_t(m,n)$, with reduced variance $w_t(m,n)\sigma_v^2$ ($< \sigma_v^2$). In that case, more noise is allowed to pass through the filter in high spatial activity regions. The weighting factor w_t is defined by:

$$w_i(m,n) = \begin{cases} 1 & \text{if } \sigma_f^2(m,n) = 0 \\ c & \text{if } \sigma_f^2(m,n) > 0, \end{cases} \quad (2.47)$$

where $0 \leq c < 1$. We propose the following weighting function:

$$w_i(m,n) = \exp[-K_1 \sigma_f^2(m,n)], \quad (2.48)$$

where K_1 is a constant to be determined by experiments (K_1 is chosen to give the best quality of restored images among different values).

Equation (2.48) looks like the so-called noise visibility function (Anderson and Netravali, [4]). However, equation (2.48) is obtained from the mere consideration of edge preservation and is not based on any subjective tests. For most noisy images, the use of w_i alone is sufficient, since the human visual system is less sensitive to noise around edges than in flat regions (Netravali and Brasada, [39]). However, it is desirable to smooth out the noise around the edges, in some cases.

As stated before, the local statistics (i.e. mean and variance) are calculated using a window of N_D^2 pixels with $f(m,n)$ being at the window's geometrical center. In the case where an edge is passing through the window, the local statistics information may be more precisely estimated, provided we could determine on which side of the edge the pixel $f(m,n)$ is located, and hence determine the orientation of the edge. The Kalman filter will be effective even along the edge.

In Figure 2.6, pixel $f(m,n)$ is located in the unshaded area which is a subset of the window D . In our case we are concerned with determining the subset and calculate the local statistics over that subset. This procedure is called refined local statistics measurement (Lee, [32]).

In order to determine the subset where $f(m,n)$ belongs, it is necessary to identify the edge orientation. To do this, we calculate the mean m_{ij} of each subregion R_{ij} , Figure 2.7. We have nine subregions with their corresponding means, Figure 2.8.

Figure 2.6 An edge across the window region.

Figure 2.7 The window D divided into subregions R_{ij} .

m_{11}	m_{12}	m_{13}
m_{21}	m_{22}	m_{23}
m_{31}	m_{32}	m_{33}

(a)

(b)

Figure 2.8 - (a) Local means for the subregions in the window
(b) Edge orientation map.

The pixel $f(m,n)$ is in subregion R_{22} . We also depicted four different orientations. Since each region has its complement we have four pairs of directions.

After $\{m_{ij}\}$ have been computed, a set of simple compass gradient masks is convolved with the mean values to determine the edge orientations. These masks are described by:

1	1	0	1	1	1	0	1	1	-1	0	1
1	0	-1	0	0	0	-1	0	1	-1	0	1
0	-1	-1	-1	-1	-1	-1	-1	0	-1	0	1

These masks are consistent with Figure 2.8. The mask which will have the maximum gradient will be the edge orientation. For example, if $|m_{11} - m_{33}| + |m_{12} - m_{23}| + |m_{21} - m_{32}|$ is the maximum, then the first direction is chosen, and we are left with determining the subset to which $f(m,n)$ belongs. For instance, $|m_{11} - m_{22}| < |m_{22} - m_{33}|$ indicates that the pixel $f(m,n)$ is more likely in the unshaded area, and the local mean and local variance of $f(m,n)$ are defined over that subset.

2.8 Simulation and Results

In the image model (2.6), the modeling coefficients are normally not known, and must be estimated. Using the least-squares algorithm, we determine the model parameters, the driving noise intensity, and the initial covariances. If A is the parameter vector, and F is the signal vector, the structures of A and F depend on specific scanning operation and the neighborhood set Ω used for estimation. Equation (2.6) is causal, and the original image is also assumed an ergodic random field.

In many cases we may have or assume some knowledge of the image statistics for example. Many experiments have shown that many images can be regarded as

wide-sense stationary random fields with a separable exponential autocorrelation function (Franks, [11]).

The following image model is used which takes the four closest pixels as neighbors, Figure 2.5. The image model (2.6) can be written explicitly as,

$$\begin{aligned} f(m,n) = & a_1 f(m-1,n) + a_2 f(m,n-1) + a_3 f(m-1,n-1) \\ & + a_4 f(m+1,n-1) + w_a(m,n) \end{aligned} \quad (2.49)$$

The experiment is carried out through four steps using several images to test the proposed sequential Kalman filter. These steps are:

- (1) estimating the modeling coefficients $\{a_i\}$;
- (2) choosing the constant K_1 in equation (2.48) for the weighting function. In the experiments, several different values of K_1 were used (between 150 and 400) using subjective judgment we determine the value of K_1 for each image used;
- (3) estimating the observed image using equations (2.3), (2.6), (2.9), (2.10), (2.26), (2.31), and (2.41);
- (4) preserving the edges using the local statistics.

To show the effectiveness of the proposed adaptive sequential Kalman filter, two standard images of size 512x420, relative to 8 bits per pixel in resolution, were tested. These images were corrupted by additive white Gaussian noise with various intensities. Figure 2.9a is the original image of the girl. Figure 2.9b-d are the corrupted versions of Figure 2.9a with noise variance $\sigma_v^2 = 400, 900, \text{ and } 1600$ respectively. The second original image, Figure 2.10a, is a collection of vegetables. The corrupted images of Figure 2.10a are shown in Figure 2.10b-d, with noise intensities of 400, 900, and 1600, respectively.

The results of the adaptive restoration procedure are shown in Figure 2.11. Figure 2.11a-c are the restored images of Figure 2.9b-d, and Figure 2.11d-f are the restored images of Figure 2.10b-d, respectively. Comparing the images of Figure 2.11 and the noisy images in Figure 2.9 and 2.10, it can be seen that the noise is suppressed and the resulting images have better quality. The adaptive Sequential Kalman filter performs well in flat regions as well as around the edges. The overall images are smooth with little blurring, which is mainly due to the particular modelling of the image field. The edges are sharp and the details of the images are preserved with a better contrast.

Figure 2.9 - Image of the girl.

- (a) The original image of the girl. (b) The corrupted image of the girl with additive white Gaussian noise with zero mean and variance $\sigma_v^2=400$. (c) The corrupted image of the girl with additive white Gaussian noise with zero mean and variance $\sigma_v^2=900$. (d) The corrupted image of the girl with additive white Gaussian noise with zero mean and variance $\sigma_v^2=1600$.

a

b

c

d

Figure 2.10 - The vegetables image.

(a) The original image. (b) The corrupted image of (a) with additive white Gaussian noise with zero mean and variance $\sigma_v^2=400$. (c) The corrupted image of (a) with additive white Gaussian noise with zero mean and variance $\sigma_v^2=900$. (d) The corrupted image of (a) with additive white Gaussian noise with zero mean and variance $\sigma_v^2=1600$.

a

b

c

d

Figure 2.11 - The restored images.
(a)-(c) The restored images of Figure 2.9 (b)-(d), respectively. (d)-(f) The restored images of Figure 2.10 (b)-(d), respectively.

a

d

b

e

c

f

Chapter 3

ROBUST MODELLING FOR IMAGE RESTORATION

3.1 Introduction

In image processing, extensive efforts have been devoted to the mathematical aspects of the problem. Modelling the image field and the degradation phenomena appears to be the most successful approach in image processing tasks, such as edge detection, image coding, image restoration, etc.. The resulting model based techniques were the most widely used methods to treat realistic images since they possess the mathematical justification and elegance. However, in all of these model based techniques the image intensity array is assumed to be a multivariate Gaussian distribution. The Gaussian assumption is used primarily in estimating the parameters of the image model fitted to the image. The corresponding estimation procedure is relatively easy, and the resulting mathematical derivations as straightforward. For example, under the Gaussian assumption, the maximum likelihood method is the same as the least squares method for the nonsymmetric half-plane (NSHP)

autoregressive model. Unfortunately, least squares estimators or maximum likelihood estimators under the Gaussian assumption are very sensitive to minor deviations from the Gaussian noise assumption. Even a single bad datum (outlier) among 1000 observations can cause large errors in the estimator. Therefore, when the image contains impulse noise, the parameter estimates obtained from the Gaussian model do not appear to be appropriate. Because of the excessive sensitivity of these commonly used estimators, robust estimators are needed in image models. A robust estimator should possess the following properties [27]:

- 1) It should have a reasonably good efficiency (optimal or nearly optimal) at the assumed noise distribution.
- 2) It should be robust in the sense that the degradation in performance caused by a small fraction of outliers is relatively small.
- 3) Somewhat larger deviations from the assumed distribution should not cause a catastrophe.

Many different robust estimation algorithms have been developed in the last two decades. These robust estimation algorithms could be classified into three large types of estimators: M-estimators, L-estimators, and R-estimators. An M-estimator is a maximum likelihood-type estimator which is obtained by solving a minimization problem. An L-estimator is a linear combination of ordered statistics. An R-estimator is derived from the rank tests.

Image restoration in the presence of noise is a fundamental problem in image processing. Therefore, robust procedures are needed. In many, if not most of the restoration techniques, the image field is considered as a multivariate Gaussian model for the convenience of mathematical derivation, and much of the effort is devoted to modelling the image field. Little attention has been given to the observation model. In most cases, the observation noise is assumed to be Gaussian. However, the Gaussian model is not appropriate for parameter estimation in the presence of outliers, and the simple use of the least-squares method is not efficient. Kashyap and Eom [27] assumed that the noise can be modeled by a contaminated Gaussian noise.

They introduced a robust restoration algorithm which treats effectively this type of impulsive noise.

In this chapter, we will introduce the robust modelling technique proposed by Kashyap and Eom [27]. We will modify the notation to be consistent with the previous chapter. Further to this, the restoration algorithm is implemented and tested to show its effectiveness.

3.2 Robust modelling

As we mentioned in the previous section, the Gaussian model is not appropriate for parameter estimation in the presence of impulse noise. However, it is more appropriate to model the noise such that outliers could be included in the distribution. Kashyap and Eom [27] assumed that the noise can be modeled by a contaminated Gaussian noise. This is more reasonable than the white Gaussian noise assumption. They defined the observation noise as,

$$\xi (m, n) = \begin{cases} v (m, n), & \text{with probability } 1 - \beta \\ w (m, n), & \text{with probability } \beta \end{cases} \quad (3.1)$$

where $v(m,n)$ is a regular white Gaussian noise, and $w(m,n)$ is an outlier sequence where the ratio of outlier β is assumed small (less than 5 percent).

3.2.1 The Nonsymmetric Half-Plane Autoregressive Model

The image intensity, which is a two-dimensional array consisting of intensities of pixels in the image plane, is usually modeled over a nonsymmetric half-plane (NSHP). Therefore, the image field is described by an NSHP autoregressive model. For such

a model, the image plane is shown in Figure 3.1. Using this representation, let (i,j) be an index for the spatial coordinate location, and $f(i, j)$ be the intensity of the image at coordinate location (i, j) . Hence, for any position of the scanner, we define the nonsymmetric half plane Ω_- as follows:

$$\begin{aligned} \Omega_- = \{ (i,j): (i < 0 \quad \text{and } j = 0); \\ (i \text{ is arbitrary} \quad \text{and } j < 0) \}. \end{aligned} \quad (3.2)$$

Figure 3.1 The nonsymmetric half-plane.

The nonsymmetric half-plane members are indicated by x's and $\circ$ is the origin.

Let s and r be indexes for two-dimensional coordinate locations of the scanner on the image plane, it can be seen that Ω_- have the following important property: if $r \in \Omega_-$ and $s \in \Omega_-$, then $(s + r) \in \Omega_-$. We will introduce the nonsymmetric half-plane model which behaves very much like the one-dimensional time series models. Assuming that the image intensity $f(s)$, at coordinate location s , follows the nonsymmetric half-plane (NSHP), an NSHP autoregressive model can be written as

$$f(s) = \sum_{r \in \Omega_-} \theta_r f(s+r) + \alpha + \sigma \zeta(s), \quad (3.3)$$

where $\zeta(\cdot)$ is white Gaussian noise with zero mean, unit variance, and uncorrelated. The parameters θ_r are the model coefficients, σ is the standard deviation of the noise, and α represents constant gray level in the image. Since only pixels closest to each other have significant correlations [6] [52], equation (3.3) can be reduced to

$$f(s) = \sum_{r \in N_1} \theta_r f(s+r) + \alpha + \sigma \zeta(s), \quad (3.4)$$

where N_1 is a subset of Ω_- . Equation (3.4) can be written in the vector form as

$$f(s) = \theta^T F(s) + \sigma \zeta(s), \quad (3.5)$$

where θ is a parameter vector, and $F(s)$ is a vector which consists of intensities of pixels in the neighbour set N_1 and unity. It is important to note that the last element of the vector $F(s)$ is required to represent a constant gray level in the image plane. A general definition of the vector $F(s)$ could be written as

$$F(s) = \text{col.} [\{ f(s+r), r \in N_1 \}, 1]. \quad (3.6)$$

For instance, if

$$N_1 = [(-1, 0), (0, -1), (-1, -1)], \quad (3.7)$$

then the model in (3.2) can be written as follows:

$$f(i, j) = \theta^T F(i, j) + \sigma \zeta(i, j), \quad (3.8)$$

$$F(i, j) = \begin{bmatrix} f(i-1, j) \\ f(i, j-1) \\ f(i-1, j-1) \\ 1 \end{bmatrix}. \quad (3.9)$$

The model given in (3.8)-(3.9) is called a three neighbour causal autoregressive model, and this model is used in the experiments, even though the following theory is applicable to any nonsymmetric half plane autoregressive model.

3.2.2 The Cost Function

In most of the recursive techniques for image restoration applications, the cost function Q is a quadratic function representing the mean-square-error between the original image and its estimate. The most popular parameter estimation approach is the least squares method where the following Q function is minimized with respect to the parameter vector θ :

$$Q = \sum_{i,j} [f(i, j) - \theta^T F(i, j)]^2 \quad (3.10)$$

Equation (3.10) is basically the minimization of the residuals. The least squares method works perfectly under the Gaussian assumption, but in the case where an outlier observation is encountered, the residual is very large and the least squares estimator is greatly influenced by this outlier. Therefore, the least squares estimator is not robust. To insure the robustness of the estimator, the residuals are scaled by a scale factor σ , which is the noise intensity in the NSHP model (3.3). Next instead of minimizing a quadratic function, a nonquadratic function ρ is minimized. The

function ρ is a differentiable function possessing a bounded derivative, and it is symmetric about the origin with $\rho(0)=0$.

To determine the model parameters, Kashyap and Eom [27] proposed a robust estimation scheme using a nonquadratic criterion function ρ rather than a quadratic function as described in Eq. (3.10). The method proposed by Kashyap and Eom is based on the maximum-likelihood estimator (M-estimator). This estimator is characterized by the minimization of a nonquadratic function Q of the normalized residuals as follows:

$$Q(\theta, \sigma) = \frac{1}{MN} \sum_{i,j} \left[\rho \left(\frac{f(i,j) - \theta^T F(i,j)}{\sigma} \right) + \frac{1}{2} \right] \sigma \quad (3.11)$$

where (MN) is the size of the image window, and σ is the noise intensity. The M-estimators of the nonsymmetric half-plane autoregressive model are defined by the minimization of the Q function described in Eq. (3.11). This translates into solving the differential equations defined as follows:

$$\frac{\partial Q(\theta, \sigma)}{\partial \theta} = -\frac{1}{MN} \sum_{i,j} \psi \left(\frac{f(i,j) - \theta^T F(i,j)}{\sigma} \right) F^T(i,j) = 0 \quad (3.12)$$

and,

$$\frac{\partial Q(\theta, \sigma)}{\partial \sigma} = \frac{1}{2} - \frac{1}{MN} \sum_{i,j} \chi \left(\frac{f(i,j) - \theta^T F(i,j)}{\sigma} \right) = 0, \quad (3.13)$$

where $\psi(x) = \partial \rho(x)/\partial x$ and $\chi(x) = x\psi(x) - \rho(x)$, and the function ψ is continuous and bounded [27]. The variable x is the normalized residual.

There are several ψ functions that can be used and the choice of such a function is very important not only to ensure the accuracy of the final estimate but also for the fast convergence of the iterative restoration procedure. Two interesting ψ functions were proposed [27]. The first is a hard limiter type ψ function described as ψ_{HL} in Eq. (3.14). The solution of Eqs. (3.12)-(3.13) is the global minimum of Eq. (3.11) because of the monotonicity of the ψ function.

$$\psi_{HL}(x) = \begin{cases} c, & x > c \\ x, & -c \leq x \leq c \\ -c, & x < -c \end{cases} \quad (3.14)$$

where c is a constant varying between 1.5 and 2.0 for this particular function.

A second ψ function, the so called redescending ψ function (ψ_{HA}) was proposed. It is a continuous function returning to zero outside some predefined interval which justifies its name. The efficiency of the redescending ψ function is claimed to be much better than the monotone hard-limiter-type ψ function, especially for extremely heavy-tailed distributions. This function is described in Eq. (3.15). The constants a , b , and c are to be chosen by experiments. The $\text{sgn}(\cdot)$ function is equal to -1 if $x < 0$ and equals 1 otherwise.

$$\psi_{HA}(x) = \begin{cases} x, & |x| \leq a \\ a \text{sgn}(x), & a < |x| \leq b \\ \frac{(c - |x|)}{(c - b)} a \text{sgn}(x), & b < |x| \leq c \\ 0, & |x| > c \end{cases} \quad (3.15)$$

The two ψ functions described in (3.14) and (3.15) were tested in the experiments, and the simulation results show that the redescending ψ function performed better than the monotone ψ function, with $a = 2.0$, $b = 2.5$, and $c = 4.5$.

3.3 The Minimization Algorithm

The robust estimator of the NSHP autoregressive model, which minimizes the criterion function described in (3.11), has many desirable properties. It has been shown that the M-estimator, minimizing (3.11), is asymptotically Gaussian and consistent with the monotonic ψ function [27]. The computation of the M-estimator is generally difficult, since it involves the minimization of a nonquadratic function of several arguments. An efficient iterative algorithm was proposed by Kashyap and Eom [27] to minimize the cost function in (3.11). They showed that the iterative algorithm converges to the robust M-estimator with the monotonic ψ function. This algorithm can be easily implemented by a small modification to a standard least squares algorithm. The iterative procedure is initialized using the least squares estimator. Because the iterations begin with the solution which minimizes a convex function (quadratic function), the risk of converging to a local minimum is very small even with a nonmonotonic ψ function.

The robust parameter estimation algorithm developed by Kashyap and Eom can be summarized as follows: to compute the robust M-estimator of the nonsymmetric half-plane autoregressive model the criterion function $Q(\theta, \sigma)$ in (3.11) is minimized with respect to θ , and σ . This can be done in the following way. First, fix the parameter σ and change the parameter θ to obtain a guaranteed decrease of the criterion function $Q(\theta, \sigma)$. Second, fix the parameter θ and change the parameter σ to obtain a guaranteed decrease of the criterion function $Q(\theta, \sigma)$. The above two steps are repeated alternately until the difference of successive iterations is judged small enough. The iterative robust estimation algorithm described above can be summarized in the following.

Algorithm 1 (Estimation with Perfect Observation)

1) Let $\theta^{(0)}$ and $\sigma^{(0)}$ be the initial estimates (the conventional least squares estimator can be used).

2) At the k th iteration, $\theta^{(k)}$ and $\sigma^{(k)}$ are available. Compute the residual $r^{(k)}$ as,

$$r^{(k)}(i, j) = f^{(k)}(i, j) - \theta^{(k)T} F(i, j), \quad (3.16)$$

where $F(i, j)$ is defined in (3.9). Compute the censored residual $r_1^{(k)}$ as,

$$r_1^{(k)}(i, j) = \psi \left(\frac{r^{(k)}(i, j)}{\sigma^{(k)}} \right) \sigma^{(k)}. \quad (3.17)$$

3) Compute $\theta^{(k+1)}$ and $\sigma^{(k+1)}$ as follows:

$$\theta^{(k+1)} = \theta^{(k)} + \tau^{(k)}, \quad (3.18)$$

where

$$\tau^{(k)} = \left[\sum_{i,j} F(i, j) F^T(i, j) \right]^{-1} \left[\sum_{i,j} F(i, j) r_1^{(k)}(i, j) \right]. \quad (3.19)$$

and,

$$\sigma^{(k+1)2} = \frac{1}{MN} \sum_{i,j} [r_1^{(k)}(i, j)]^2. \quad (3.20)$$

4) Repeat steps 2) and 3) until the differences $\|\theta^{(k+1)} - \theta^{(k)}\|$ and $|\sigma^{(k+1)} - \sigma^{(k)}|$ become negligible.

The theoretical convergence of the iterative method was shown in [27]. The properties of the robust estimation algorithm were summarized in two Lemmas and

one theorem. Lemma 1 and Lemma 2 show the decrease of the Q function in (3.11) with the update of parameters θ and σ at each iteration. Let $Q(\theta^{(k)}, \sigma^{(k)})$ be the cost involved with the estimates $\theta^{(k)}$ and $\sigma^{(k)}$ at the kth iteration. That is,

$$Q(\theta^{(k)}, \sigma^{(k)}) = \frac{1}{MN} \sum_{i,j} \left[\rho \left(\frac{f(i,j) - \theta^{(k)T} F(i,j)}{\sigma^{(k)}} \right) + \frac{1}{2} \right] \sigma^{(k)}. \quad (3.21)$$

Lemma 1: At every iteration step, $\theta^{(k+1)}$ in (3.18) with the ψ function as in (3.14) reduces the Q function for fixed $\sigma^{(k)}$, i.e.,

$$Q(\theta^{(k)}, \sigma^{(k)}) - Q(\theta^{(k+1)}, \sigma^{(k)}) \geq \frac{1}{2\sigma^{(k)}MN} \tau^{(k)T} \left[\sum_{i,j} F(i,j) F^T(i,j) \right] \tau^{(k)} \quad (3.22)$$

Lemma 2: At every iteration step, $\sigma^{(k+1)}$ in (3.20) with the ψ function as in (3.14) reduces the Q function for fixed $\theta^{(k)}$, i.e.,

$$Q(\theta^{(k)}, \sigma^{(k)}) - Q(\theta^{(k)}, \sigma^{(k+1)}) \geq \frac{(\sigma^{(k+1)} - \sigma^{(k)})^2}{2\sigma^{(k)}} \quad (3.23)$$

The iterative estimation algorithm (Algorithm 1) decreases the value of the Q function in (3.11) as the number of iterations increases, and it converges to the minimum value of the function Q. The estimated parameters also converge to the parameters which minimize the function Q in (3.11). The above property of the algorithm is summarized in the following theorem.

Theorem 1: The sequence $\{(\theta^{(k)}, \sigma^{(k)})\}$ converges to $(\hat{\theta}, \hat{\sigma})$, a unique solution of (3.12) and (3.13), and an estimator which minimizes (3.11).

A proof of Lemmas 1, 2, and Theorem 1 can be found in [27]. The robust parameter estimation algorithm presented above is much less sensitive to outliers than the least squares algorithm because the ψ function limits the influence of outliers. The redescending ψ function performed better than the hard-limiter type ψ function. Therefore, the redescending ψ function was used in the simulations. The theoretical convergence of the robust algorithm was shown for perfect observations. However, realistic images always suffer from various degradations, and the simple use of algorithm 1 will not be justified. Therefore, the problem of parameter estimation and image recovery have to be treated simultaneously. In the following section, we introduce the restoration scheme which includes both parameter estimation and image recovery.

3.4 Parameter Estimation with Noisy Image and Image Restoration

In this section parameter estimation from an image corrupted by impulse noise is considered. In real applications, only a contaminated version of the image is available. Therefore the original image is not directly observable. We assume that the original process follows the nonsymmetric half-plane autoregressive model.

$$f(i, j) = \theta^T F(i, j) + \sigma \zeta(i, j), \quad (3.24)$$

where $\zeta(i, j)$ is a standard Gaussian process, θ is a parameter vector, and $F(i, j)$ is defined in (3.25).

$$F(i, j) = \text{col.} [\{ f(i, j) + r, r \in N_1 \}, 1] . \quad (3.25)$$

We assume that the observation $g(i, j)$ is contaminated by a noise process containing outliers. The observation $g(i, j)$ can be represented by the following equation:

$$g(i, j) = f(i, j) + \xi(i, j). \quad (3.26)$$

The noise $\xi(i, j)$ is a white Gaussian noise containing a small fraction of outliers. This was described in (3.1).

To estimate the parameters in the above model, the function given in (2.11) cannot be directly used since the original image $f(i, j)$ is not available. However, it is obvious that the problem of parameter estimation cannot be separated from the problem of image recovery. A possible solution to the recovery problem is to minimize a modified version of the Q function in (2.11) to account for the fact that the original image $f(i, j)$ is not available. Kashyap and Eom used an iterative approach that combines both robust parameter estimation and image recovery. Suppose that at the k th iteration, the estimate of the original image f is $f^{(k)}$. The function Q in (2.11) can be minimized with f replaced by $f^{(k)}$, and get the estimates $\theta^{(k)}$ of θ . Then, a method for improving the estimate of f is needed, the so-called data cleaning. The idea of data cleaning is the following: if the observation is contaminated by an outlier, the residual will be large. If the normalized residual is large, the observation is modified to reduce the effect of outliers using the ψ function. This can be summarized in algorithm 2.

Algorithm 2: (Estimation with Noisy Observation)

- 1) initially set $f^{(0)}(i, j) = g(i, j)$. Compute the initial estimate $\theta^{(0)}$ and $\sigma^{(0)}$ from the noisy observation $\{g(i, j)\}$ using the standard least squares algorithm.

- 2) Consider the k th iteration, $f^{(k)}$, $\theta^{(k)}$, and $\sigma^{(k)}$ are available. Get the updated estimate of $f(\cdot)$ from $f^{(k)}(\cdot)$ to $f^{(k+1)}(\cdot)$ by the following formula:

$$f^{(k+1)}(i, j) = \theta^{(k)T} F^{(k)}(i, j) + \psi \left(\frac{f^{(k)}(i, j) - \theta^{(k)T} F^{(k)}(i, j)}{\sigma^{(k)}} \right) \sigma^{(k)} \quad (3.27)$$

where

$$F^{(k)}(i, j) = \text{col. } [\{ f^{(k)}((i, j) + r), r \in N_1\}, 1]. \quad (3.28)$$

- 3) Obtain estimates $\theta^{(k+1)}$ and $\sigma^{(k+1)}$ from the cleaned data $f^{(k+1)}$ by minimizing the following function:

$$Q = \frac{1}{MN} \sum_{i,j} \left[\rho \left(\frac{f^{(k+1)}(i, j) - \theta^{T} F^{(k+1)}(i, j)}{\sigma} \right) + \frac{1}{2} \right] \sigma \quad (3.29)$$

this can be computed by the Algorithm 1.

- 4) Repeat steps 2) and 3) until the differences of estimates between iterations become small.

In the above developments, it was not possible to show the theoretical robustness of the algorithms. However, Kashyap and his colleague conducted extensive experiments and the results of the simulation showed the robustness of the estimator in Algorithm 2. To test the algorithms, they generated one hundred artificial images, of size 25×25 , containing outliers ranging from 0 to 10 percent of the entire image field. Both the robust estimator and the least squares estimator were applied to the images, and the parameters model were estimated. Using the three neighbour NSHP autoregressive model described in section 3.2, the simulation results show that the robust estimator is more efficient than the least squares estimator since

the mean square errors are significantly lower. The least squares is very sensitive to outliers, but the robust estimator is much less sensitive to outliers.

The only assumption made is that the entire image obeys an NSHP model. In practice, a realistic image, say 512×512 , will not obey a simple model due its structure. However, the image field can be divided into individual windows, and the image can be modelled in each window, e.g. 8×8 , by an NSHP model. These windows are applied from upper left corner to lower right corner in the raster scanning direction, but two adjacent windows overlap only for one column or one row (Figure 3.2). By overlapping one column or one row with the previously processed window, we can insure continuity of the image data at the boundary pixels and eliminate the so-called boundary effects. For each window, Algorithm 2 is applied, and the pixels inside the window, namely, the 7×7 core of the window, are replaced by the cleaned data obtained by Algorithm 2. The window size is not critical to the performance of the algorithm. A window size of 8×8 was chosen and it performed well in the experiments.

The robust algorithm is further applied to restore several noisy images. The noisy images were constructed by adding both Gaussian (0,100) noise and 5 percent of impulse noise to the original images. The impulse noise is generated to have equal probability of having gray levels 0 (black) and 255 (white). The convergence of the robust algorithm was shown by the decrease of the mean square error at each iteration. As the iterations progress the mean square error decreases considerably to stabilize after three iterations. From the experimental results [27], the robust model based restoration algorithm performed better than some of the classical techniques used to reduce impulse noise, such as the median filter, the mean filter, and the α -trimmed mean filter. These conventional methods replace each pixel by its location estimates. Because these methods are based on the constant intensity assumption, the details of the original image are significantly blurred. However, the restoration algorithm based on the robust model does not blur images.

Figure 3.2 Division of the image into small windows: one row or one column is overlapped between two adjacent windows following the raster scanning operation.

3.5 Simulation

We implemented the robust restoration procedure described above to show the effectiveness of the algorithm in suppressing impulsive noise as well as Gaussian noise. Two images of size 512x512 were used in the simulation. The original images of a girl and a baboon are shown in Figure 3.3 (a) and (b), respectively. The images are digitized into 256 gray levels. To measure the performance of the robust restoration procedure on noisy pictures, the contaminated images are constructed by

adding both Gaussian (0,100) noise and 5 percent of impulse noise to the original images. The corrupted images of the girl and the baboon are shown in Figure 3.3 (c) and (d), respectively. The impulse noise is generated to have equal probabilities of having gray level 0 (black) and 255 (white). The robust model based restoration algorithm is applied to the contaminated images. The results of the restoration procedure are shown in Figure 3.3 (e) and (f) for the girl and the baboon, respectively. These images are obtained after three iterations of the data cleaning process using the redescending ψ function. It can be seen that the impulsive noise is almost completely absent, and the residual Gaussian noise is hardly noticeable. Therefore, it is clear that the robust algorithm is able to suppress both Gaussian noise as well as impulsive noise. The edge preserving capability of the algorithm can be seen. The fine details of the original images are well shown in the restored images. For example, the eyes and the mouth of the girl's face [Figure 3.3 (e)], and the hair of the baboon's face [Figure 3.3 (f)] are well preserved and have sharp edges.

Since the robust algorithm performed well in suppressing both Gaussian noise and impulsive noise, we applied the algorithm to images that are contaminated by Gaussian noise only, no outliers present. Two images were used in the experiments, Figure 3.4. Figure 3.4 (a) and (b) show the 512x512 images of a vegas girl and a square shaped pattern relative to 8-bits per pixel in resolution. These images were contaminated by additive Gaussian noise with zero mean and variance 625. The corrupted images of the vegas girl and the square pattern are shown in Figure 3.4 (c) and (d), respectively. We applied the robust model based algorithm to these images, and the results of the restoration are shown in Figure 3.4 (e) and (f), respectively. Three iterations and the redescending ψ function are used. The results show that the robust model is working effectively in suppressing the noise. The edges are well preserved and the resulting images are sharp with a good visual quality.

The successful results obtained by the present robust modelling approach could be the basis for more research in applying robust techniques in image processing tasks, such as image coding, image restoration, edge detection, etc.. However, many of the restoration techniques based on the Gaussian assumption can

be improved to suppress impulsive noise. In the following chapter, we will investigate the use of robust modelling in the Kalman filtering procedure.

Figure 3.3 The girl and baboon pictures.

(a) the original image of the girl. (b) The original image of the baboon. (c) and (d) The corrupted images of the girl and the baboon, respectively. (e) and (f) The restored images of the girl and the baboon using the robust model based restoration algorithm. Only three iterations are used.

a

b

c

d

e

f

Figure 3.4 The vegas and the square pattern images.

(a) (b) The original 512x512 images of the vegas and the square shaped pattern, respectively. (c) and (d) The degraded images of the vegas and the square pattern, respectively. These are obtained by adding Gaussian noise with variance $\sigma_v^2=625$ to the originals in (a) and (b). (e) and (f) The restored images of (c) and (d) using the robust restoration algorithm.

a

b

c

d

e

f

Chapter 4

A MODIFIED REDUCED UPDATE KALMAN FILTER FOR IMAGE RESTORATION: ROBUST MODELLING

4.1 Introduction

One of the inherent problems, in model based techniques, for the restoration of noisy images, is the blurring and smearing of edges which carry most of the details in the image itself. Kalman filtering is one of the estimators that has been used for restoration with some degree of success. However, the artifacts present around the edges are still pertinent and the ill-posed modelling problem is one of the main sources for the Gaussian noise assumption. It has been shown, chapters 1 and 2, that the blurring could be reduced by modelling the image field by a nonstationary field.

The local statistical characteristics of the image are used to preserve the edges, but this is done at the expense of noise removal around the edges.

However, in all model based techniques, the image intensity array is assumed to be a multivariate Gaussian distribution. The Gaussian assumption is primarily used in estimating the parameters of the image model fitted to the image. This assumption simplifies the mathematics of the procedure used, but the Gaussian model is not appropriate for parameter estimation in the presence of impulse noise in the image, and the simple use of the least-squares method will not be efficient. Therefore, more appropriate means of determining the model parameters have to be established. Robust modelling techniques are the most commonly used methods to reduce the effect of impulsive noise, i.e. the median filter and the α -trimmed mean filter. An effective robust estimation procedure was developed in the previous chapter which appears to be more efficient than the classical methods used to treat impulsive noise.

Developments of Kalman-type filters are based on the Gaussian assumption for the noise and the image intensity array. The robustness of the Kalman estimator has been investigated by many researchers in the field. For example, Bryson and Ho [8] have shown that the Kalman filter is the solution to the weighted-least-squares estimator problem, and Nahi [36] has shown that the Kalman filter is the best recursive linear estimator in the sense of minimizing the expected weighted-squared-error. These examples show the insensitivity of the filter to cost function variation or to restrictions on the class of admissible estimators. Ho [19] insured the robustness argument by showing that the Kalman filter algorithm is a "good" estimator in some problems where the assumptions of linear system structure and Gaussian noise processes are not valid. Morris [34] showed that there exists a Kalman filter which is a minimax estimator against a wide class of driving noise, measurement noise and initial state distributions for a linear system model and the expected square-error cost function which indicates additional robustness properties of the Kalman filter and provides a theoretical basis for widespread engineering practice. It has been shown [27] that the square error criterion, used for parameter estimation is not appropriate in the presence of impulse noise. However, the robust estimation procedure

developed in the previous chapter could be incorporated into the Kalman filter to restore images contaminated by impulse noise.

4.2 The Reduced Update Kalman Filter

In contrast to the sequential Kalman filter presented in chapter 2, in this chapter a Reduced Update Kalman filter (RUKF) will be implemented. The difference between the two methods is that the RUKF updates five pixels simultaneously whereas the sequential Kalman filter updates one pixel at a time. Due to the recursive nature of the Kalman filter, the RUKF estimates each pixel five times as the scanner moves from the left side of the image to the right end. The difference in performance between the reduced update Kalman filter and the sequential Kalman filter have not been evaluated.

Woods and Radewan [52] conducted an extensive survey about two-dimensional Kalman filters. They developed an approximation to the scalar Kalman processor, called the reduced update Kalman filter (RUKF). The state vector for this filter is (MN) -dimensional, where M is the order of the filter and N is the width of the picture. The main concept of the reduced update Kalman filter is that the updating process in the Kalman equations is confined to a small portion of the state vector. By updating only those pixels close to the one being processed, the amount of computations is significantly reduced and the restoration procedure is nearly optimal.

The image intensity array, consisting of intensities of the pixels in the image plane, is usually modeled over a nonsymmetric half-plane (NSHP). Therefore, the image field is described by an NSHP autoregressive model. For such a model, the image plane is shown in Figure 2.1. Using this representation, for any position of the scanner, all the previously processed pixels are considered as "past" states, while the pixel which is presently being processed is the "present" state, and those picture

elements to be processed are referred to as the "future" states. Obviously, these definitions are rather artificial as the scanning operation can take any order following any particular direction.

Let (m,n) be the present spatial position of the scanner on the image plane. Assuming that the image intensity $f(m,n)$, at location (m,n) , follows the nonsymmetric half-plane (NSHP), we define $P(m,n)$ similar to Eq.(2.1) as the past region with respect to location (m,n) ,

$$P(m,n) = \{(i,j): \{1 \leq i \leq N, 1 \leq j \leq n-1\} \cup \{1 \leq i \leq m-1, j=n\}\} \quad (4.1)$$

where N is the width of the image plane for a raster scanning operation. The nonsymmetric half-plane members are those satisfying Eq. (4.1). The above definition of the nonsymmetric half-plane is similar to the definitions in (2.1) and (3.2), and this will be used throughout the entire developments below.

Figure 4.1 The nonsymmetric half-plane (NSHP).

Since the concept of state is central to linear system theory, Woods and his colleague defined the state as the minimum amount of information about the past and present estimates required to determine the optimal causal estimate of the future response given the noisy observations. If the filter is of order $(M \times M)$ the state vector will be defined over a portion of the nonsymmetric half plane $P(m,n)$ and has $O(MN)$ components, where M is the order of the recursive model and N is the width of the image. The support of the NSHP model is called the pseudo-state vector, and the update process is conducted over the $O(M^2)$ points for each observation.

Following the nonsymmetric half-plane, the image signal at (m,n) can be modelled by the NSHP autoregressive model as,

$$f(m, n) = \sum_{(k,l) \in P(m,n)} c_{kl} f(m-k, n-l) + w(m, n) \quad (4.2)$$

where $\{w(m,n)\}$ is zero mean homogeneous white Gaussian source with variance σ_w^2 . The c_{kl} 's are the model parameters, and $f(m,n)$ is the image intensity. The scalar observation is given by

$$g(m,n) = f(m,n) + v(m,n), \quad (4.3)$$

where $\{v(m,n)\}$ is a zero mean homogeneous white Gaussian noise, independent of $\{w(m,n)\}$, with variance σ_v^2 . Woods and Radewan [52], defined the state vector $F(m,n)$ as,

$$F(m,n) = [f(m,n), f(m-1,n), \dots, f(1,n); f(N,n-1), \dots, f(1,n-1); \dots; f(N,n-M), \dots, f(m-M,n-M)]^T. \quad (4.4)$$

The elements of the state vector F are the intensities of pixels at the corresponding coordinates. The pseudo-state vector or the support of the NSHP model is given by:

$$F_1(m,n) = [f(m,n), f(m-1,n), \dots, f(m-M,n); f(m+M,n-1), \dots, f(m-M,n-1); \dots; f(m+M,n-M), \dots, f(m-M,n-M)]^T. \quad (4.5)$$

The remainder of the state vector $F(m,n)$ are ordered into $F_2(m,n)$. Thus the resulting assignment of points is as shown in Figure 4.2. With this convention, the state vector can be written as,

$$F(m,n) = [F_1(m,n), F_2(m,n)]^T. \quad (4.6)$$

The state dynamical model can be written in the vector form as

$$F(m,n) = CF(m-1,n) + w(m,n), \quad (4.7)$$

Figure 4.2 The partitioning of the state vector.

where C is the system propagation matrix determined from the model parameters and the ordering of the state vector $F(m,n)$. It can be noted that Eq. (4.7) holds for all points (m,n) except near the boundaries of the image, where boundary conditions have to be incorporated. The drive vector is given as

$$w(m,n) = [w(m,n), 0, \dots, 0]^T, \quad (4.8)$$

and the scalar observation equation is

$$r(m,n) = HF(m,n) + v(m,n), \quad (4.9)$$

where

$$H = [1, 0, \dots, 0]. \quad (4.10)$$

The matrix C is partitioned similarly to F , which can be defined in the following form

$$C = \begin{pmatrix} C_{11} & C_{12} \\ C_{21} & C_{22} \end{pmatrix} \quad (4.11)$$

For this particular partitioning of C , it turns out that C_{11} and C_{12} contain all the $\{c_{kl}\}$ terms, and the remainder of C is only a shift operation containing zeros and ones. Equation (4.7) can be written as,

$$F_1(m,n) = C_{11}F_1(m-1,n) + w_1(m,n) + C_{12}F_2(m-1,n), \quad (4.12)$$

where w has been partitioned similarly to F . It can be seen that F_2 requires only the shifting of previously computed values. The vector H can also be partitioned to $H=[H_1, H_2]$ with $H_2 = 0$ to get a new observation equation

$$g(m,n) = H_1 F_1(m,n) + v(m,n). \quad (4.13)$$

The Kalman gain is partitioned similarly to F where $K(m,n) = [K_1(m,n), 0]^T$. The vector 0 is a null vector of proper dimension. Therefore, the Kalman equations can be written in the vector form as,

1) extrapolation: $m \rightarrow m + 1$

$$P_b(m,n) = C P_a(m-1,n) C^T + G Q_u G^T \quad (4.14)$$

where $Q_u = E[ww^T]$, which is the covariance of the drive noise.

$$F_{1b}(m,n) = C_{11} F_{1a}(m-1,n) + C_{12} F_{2a}(m-1,n). \quad (4.15)$$

2) update:

$$K_1(m,n) = P_{11,b}(m,n) H_1^T (H_1 P_{11,b}(m,n) H_1^T + \sigma_v^2)^{-1}, \quad (4.16)$$

$$F_{1a}(m,n) = F_{1b}(m,n) + K_1(m,n) [g(m,n) - H_1 F_{1b}(m,n)], \quad (4.17)$$

$$P_{11,a}(m,n) = [I - K_1(m,n) H_1] P_{11,b}(m,n), \quad (4.18)$$

$$P_{12,a}(m,n) = [I - K_1(m,n) H_1] P_{12,b}(m,n). \quad (4.19)$$

The subscripts b and a represent before and after updating, respectively. In this case, the matrix G is the unit matrix. The matrix P is partitioned similarly to F which is defined as the covariance of the estimation error. The algorithm described in Eqs. (4.14)-(4.19) is the vector form of the sequential Kalman filter discussed in chapter 2. The only difference is in the structure of the model matrices.

The reduced update Kalman filter was developed as an optimal approximation to the 2-D scalar Kalman filter. It was shown to be an optimal filter in that it minimizes the post update mean square-error under the constraint of updating only the nearby previously processed neighbours. Woods and Radewan [52] showed that for an all-pole data model, the reduced update Kalman filter converges to an optimum 2-D recursive filter in the homogeneous case. However, when the noise is not homogeneous and does not follow a white Gaussian distribution, the resulting estimate is not optimal and the possibility of convergence is very slim. A second disadvantage of the RUKF is the blurring and the artifacts present around the edges. This problem is apparent in linear system models. The edges are smoothed out and the restored images have lower contrast.

In the Kalman filter approach, the model parameters are obtained prior to the restoration scheme. The least squares is usually used to estimate these parameters. However, the least squares is not an effective parameter estimation method in the presence of impulse noise as we mentioned in the previous chapter. Therefore, we cannot separate the problem of parameter estimation from the problem of image recovery. The Kalman procedure has to be modified such that the model parameters and the image can be estimated alternately.

With these problems encountered in Kalman filtering applications, the need for better quality images led to the development of a new Kalman filtering technique using a modified reduced update Kalman filter. In the following section, we develop a restoration algorithm that uses robust parameter estimation, combined with the RUKF method, to restore images corrupted by a mixture of impulse noise and Gaussian noise.

4.3 New Development

In the previous section, the reduced update Kalman filter was developed to restore images from their noisy counterparts. The noise was assumed to be a homogeneous white Gaussian noise, and the model parameters are obtained prior to the restoration procedure. With the Gaussian noise assumption, the least squares method is usually used to estimate the parameters, and the problem of parameter estimation is treated separately from the problem of image recovery. However, if the noise is impulsive, parameter estimation and image restoration cannot be separated and have to be treated simultaneously. To do so, we have to combine both robust parameter estimation and Kalman filtering. Therefore, we model the image such that both techniques can be implemented alternately.

First we model the image field by a nonsymmetric half-plane autoregressive model. Following the nonsymmetric half-plane $P(m,n)$ in (4.1), the image intensity at (m,n) can be modelled by an autoregressive model as,

$$f(m,n) = \sum_{(i,j) \in P(m,n)} a_{ij} f(m-i, n-j) + \alpha + \sigma \zeta(m,n) \quad (4.20)$$

where $\zeta(m,n)$ is the modelling error which is assumed to be Gaussian and white with zero mean, unit variance, and uncorrelated. The a_{ij} 's are the model parameters, $f(m,n)$ is the image intensity, and α represents a constant grey level in the image. Since only the close neighbours of the pixel at location (m,n) are the most significant in modelling $f(m,n)$, we model the image intensity over a subregion Ω_1 . Therefore, equation (4.20) can be reduced to

$$f(m,n) = \sum_{(i,j) \in \Omega_1} a_{ij} f(m-i, n-j) + \alpha + \sigma \zeta(m,n) \quad (4.21)$$

By choosing only four neighbours to the pixel at location (m,n) , the model in (4.21) is called a four neighbour autoregressive model which can be written in the vector form as,

$$f(m,n) = A^T z(m,n) + \sigma \zeta(m,n), \quad (4.22)$$

where A is the parameter vector, and $z(m,n)$ is a vector of intensities in the neighbour set Ω_1 and unity. The vector $z(m,n)$ can be written explicitly as

$$z(m,n) = [f(m-1,n) \ f(m+1,n-1) \ f(m,n-1) \ f(m-1,n-1) \ 1]^T. \quad (4.23)$$

The observation model can be written in a scalar form similar to equation (3.26) As,

$$g(m,n) = f(m,n) + \xi(m,n) \quad (4.23)$$

where the noise $\xi(m,n)$ is a Gaussian noise containing outliers. ξ is defined similar to equation (3.1) as,

$$\xi(m,n) = \begin{cases} v(m,n) & \text{with probability } 1 - \beta \\ u(m,n) & \text{with probability } \beta \end{cases} \quad (4.24)$$

where $v(m,n)$ is a regular white Gaussian noise, and $u(m,n)$ is an outlier sequence which occupies β percent of the entire image field.

Realistic images usually have complex structures and the image intensity array is of high dimensionality. Therefore, the entire image field will not obey a single model. However, we can divide the image field into small windows, say 8×8 , where the image can be modelled by an NSHP autoregressive model. By dividing the image into windows, we can achieve computational and storage efficiency. Due to this partitioning of the image field, we have to insure the continuity of the image data at

the edges of the window. This can be done by overlapping adjacent windows following the raster scanning operation [Figure 3.2].

We proceed by estimating the parameters of the model, namely the parameter vector A and the noise variance. The parameter estimation algorithm is initialized by calculating the initial parameter vector $A^{(0)}$ using the least-squares algorithm. The initial noise variance is obtained as follows

$$\sigma^2^{(0)} = \frac{1}{D} \sum_{m,n} [g(m,n) - A^{(0)T} z^{(0)}]^2. \quad (4.25)$$

We set the initial image estimate to the actual observation [$f^{(0)}(m,n) = g(m,n)$]. But, since the robust algorithm developed in the previous chapter, Algorithm 2, proved to be more efficient for parameter estimation than the least-squares method, we improve the parameter estimates by conducting the restoration procedure as follows:

- 1) Compute the parameter vector $A^{(0)}$ using the least squares. Compute the noise intensity $\sigma^{(0)}$ using Eq. (4.27).
- 2) Compute an estimate of $f(m,n)$ from $f^{(p)}(m,n)$ to $f^{(p+1)}(m,n)$ by an equation similar to Eq.(3.27). That is

$$f^{(p+1)}(m,n) = A^{(p)T} z^{(p)}(m,n) + \psi \left(\frac{f^{(p)} - A^{(p)T} z^{(p)}(m,n)}{\sigma^{(p)}} \right) \sigma^{(p)}. \quad (4.26)$$

- 3) Update both the parameter vector and the noise intensity by minimizing the

following function:

$$Q(A, \sigma) = \frac{1}{D} \sum_{m,n} \left[\rho \left(\frac{f^{(p+1)}(m,n) - A^T z^{(p+1)}(m,n)}{\sigma} \right) + \frac{1}{2} \right] \sigma. \quad (4.27)$$

The function ρ is a nonquadratic function of the normalized residual. It is a differentiable function possessing a bounded derivative, and is symmetric about the origin with $\rho(0) = 0$. The ψ function in equation (4.26) is defined by $\psi(x) = \partial \rho(x) / \partial x$. More details about these functions are given in the previous chapter. The criterion function in (4.27) can be minimized by the following iterative procedure:

a) At the k th iteration, $A^{(k)}$ and $\sigma^{(k)}$ are available. Compute the residual $r^{(k)}$

$$r^{(k)}(m,n) = f^{(p+1)}(m,n) - A^{(k)T} z^{(p+1)}(m,n). \quad (4.28)$$

where $f^{(p+1)}(m,n)$ and $z^{(p+1)}(m,n)$ are obtained from the cleaned data. Compute the censored residual $\hat{r}^{(k)}$.

$$\hat{r}^{(k)}(m,n) = \psi \left(\frac{r^{(k)}(m,n)}{\sigma^{(k)}} \right) \sigma^{(k)}. \quad (4.29)$$

b) Compute $A^{(k+1)}$ and $\sigma^{(k+1)}$ as follows:

$$A^{(k+1)} = A^{(k)} + \hat{r}^{(k)}, \quad (4.30)$$

where

$$\tau^{(k)} = \left[\sum_{m,n} z(m,n) z^T(m,n) \right]^{-1} \left[\sum_{m,n} z(m,n) \hat{r}^{(k)}(m,n) \right] \quad (4.31)$$

with $z(m,n)$ being obtained from the cleaned data, where $z(m,n)$ is defined by $z(m,n) = z^{(p+1)}(m,n)$.

$$\sigma^{(p+1)^2} = \frac{1}{D} \sum_{m,n} [\hat{r}^{(p)}(m,n)]^2. \quad (4.32)$$

c) Repeat steps a) and b) until the differences $\|A^{(k+1)} - A^{(k)}\|$ and $|\sigma^{(k+1)} - \sigma^{(k)}|$ become negligible.

The final estimates obtained by the above procedure are the updated estimates of A and σ^2 . The ψ function used in the above procedure is the redescending ψ function described in Eq. (3.15). It is important to note that the noise variance σ^2 is actually a combination of the driving noise σ_w^2 and the observation noise σ_v^2 in the Kalman equations. Therefore, the updated noise variance can be written as,

$$\sigma^2 = \sigma_w^2 + \sigma_v^2 \quad (4.33)$$

It has been shown [27] that the resulting noise distribution is asymptotically Gaussian. The approximate Gaussian distribution is an immediate result of using the ψ function which limits the influence of outliers. The initial observation noise intensity σ_v^2 is set to the intensity of the Gaussian part in Eq. (4.8). By doing so, the driving noise intensity can be obtained by:

$$\sigma_w^2 = \sigma^2 - \sigma_v^2 \quad (4.34)$$

Now that all the necessary parameters are available, we can estimate the image intensity using the Kalman filter equations (4.14)-(4.19). First, we obtain the initial covariance matrix from the cleaned data obtained from the first iteration. Therefore, we make sure that the Kalman estimator will not diverge since the initial conditions are well established. The elements of the parameter vector A are ordered into the propagation matrix C in Eq.(4.11) such that the Kalman procedure can be implemented properly. The updating process is thus confined to the support of the pseudo-state vector which is a subregion Ω including $f(m,n)$ and the four closest neighbours contained in the vector $z(m,n)$ described in (4.23).

Having determined the initial conditions and the initial parameters, we develop an iterative Kalman processor which is a modified version of the reduced update Kalman filter described in section 4.2. The objective of such technique is to find an estimate $f^{(p+1)}(m,n)$ of $f(m,n)$ such that

$$f^{(p+1)}(m, n) = \hat{f}_a(m, n) = \hat{f}_b(m, n) + K(m, n) e(m, n) \quad (4.35)$$

where p is the iteration step. The subscripts a and b represent after and before updating, respectively. The term $e(m,n)$ is the residual at the location (m,n) , which is often called the innovation process. This plays an important role in recursive filtering techniques, and is defined as

$$e(m,n) = g(m,n) - \hat{f}_b(m,n), \quad (4.36)$$

where $\hat{f}_b(m,n)$ is the predicted estimate of $f(m,n)$. $K(m,n)$ in (4.35) is the Kalman gain which is determined by the error covariance, the noise statistics, and the model parameters.

In the following development, we present the modified Kalman equations and the steps involved in the procedure. These steps are the prediction step and the updating step similar to the so-called data cleaning algorithm [27]. The prediction step consists of determining the predicted estimates of the image intensity and the covariance matrix. The update process improves the predicted estimates.

a) *Prediction:*

The predicted estimate at (m,n) is given by:

$$\hat{f}_b(m, n) = \sum_{\mathbf{a}} c_{\mathbf{a}} \hat{f}_a(m-k, n-l). \quad (4.37)$$

The predicted error covariance can be obtained by an equation similar to (4.14). This is described in the scalar form as

$$R_b(m, n; k, l) = \sum_{\mathbf{o,p}} c_{\mathbf{o,p}} R_a(m-o, n-p; k, l), \quad (k,l) \in S \quad (4.38)$$

and,

$$R_b(m, n; m, n) = \sum_{k,l} c_{k,l} R_b(m, n; m-k, n-l) + \sigma_w^2, \quad (4.39)$$

Equations (4.38) and (4.39) correspond to Eq. (4.14), where σ_u^2 is the drive noise intensity. The set S is the support of the state vector.

b) *Update:*

The Kalman gain is obtained from equation (4.16) as follows

$$K(i, j) = \frac{R_b(m, n; i, j)}{R_b(m, n; m, n) + \sigma_v^2}, \quad (i, j) \in \Omega \quad (4.40)$$

and the updated covariance matrix is given as

$$R_a(i, j; k, l) = R_b(i, j; k, l) - K(m-i, n-j) R_b(m, n; k, l),$$

for $(i, j) \in \Omega; (k, l) \in S$. (4.41)

The updated estimate of the image over the subregion Ω , at coordinate position (m, n) , is given as

$$\hat{f}_a(i, j) = \hat{f}_b(i, j) + K(i, j) e(m, n), \quad (i, j) \in \Omega. \quad (4.42)$$

The cleaned data is obtained from the updated estimates of the image, namely $f^{(p+1)}(m, n) = \hat{f}_a(m, n)$. The variance of the observation noise at iteration step $(p+1)$ is obtained by the following equation,

$$(\sigma_v^{(p+1)})^2 = \frac{1}{D} \sum_{m,n} [\hat{e}_1^{(p)}(m, n)]^2. \quad (4.43)$$

where $\hat{e}_1^{(p)}$ is the new residual computed from the cleaned data and the predicted estimate. This is defined as,

$$\hat{e}_1^{(p)}(m, n) = \psi \left(\frac{\hat{f}_a(m, n) - \hat{f}_b}{\sigma_v^{(p)}} \right) \sigma_v^{(p)}, \quad (4.44)$$

The constant D is the size of the image window. Once the entire window is scanned and the estimate $f(p+1)$ is obtained, we can update the parameter vector A , from $A^{(p)}$ to $A^{(p+1)}$, and the driving noise σ_w , from $\sigma_w^{(p)}$ to $\sigma_w^{(p+1)}$, similar to the above robust procedure given in steps a), b), and c), and described by Eqs. (4.26)-(4.32). This can be done iteratively as follows:

i) at the k th iteration, $A^{(k)}$ and $\sigma^{(k)}$ are available with $A^{(k)} = A^{(p)}$, and $\sigma^{(k)} = \sigma^{(p)}$. Compute the residual $e^{(k)}$

$$e^{(k)}(m, n) = \hat{f}_a(m, n) - A^{(k)T} z(m, n), \quad (4.45)$$

The censored residual $\hat{e}^{(k)}(m, n)$ can be calculated as follows,

$$\hat{e}^{(k)}(m, n) = \psi \left(\frac{e^{(k)}(m, n)}{\sigma_w^{(k)}} \right) \sigma_w^{(k)}. \quad (4.46)$$

The choice of the ψ function is determined by experiments. However, the present algorithm performed best with the redescending ψ function (3.15), with $a=1.5$, $b=2.5$,

and $c=3.5$. The choice of these values is justified by the fact that, for a Gaussian distribution, more than 98% of the noise values, at any location (m,n) , are between $\pm 3\sigma$. Now, since we are using the censored residual instead of the actual one, the bounded ψ function will limit the influence of the outlier sequence. Hence, the intensity of the observation noise is controlled and the resulting distribution is asymptotically Gaussian, leading to an approximately Gaussian distribution.

ii) Update the model parameter vector and the variance of the driving noise as follows,

$$A^{(k+1)} = A^{(k)} + T^{(k)}, \quad (4.47)$$

where,

$$T^{(k)} = \left[\sum_{m,n} z(m, n) z^T(m, n) \right]^{-1} \left[\sum_{m,n} z(m, n) \hat{e}^{(k)}(m, n) \right]. \quad (4.48)$$

The variance of the driving noise is updated using the following equation

$$(\sigma^{(k+1)})^2 = \frac{1}{D} \sum_{m,n} [\hat{e}^{(k)}(m, n)]^2. \quad (4.49)$$

iii) repeat steps i) and ii) until the difference between estimates is negligible.

Equations (4.37)-(4.49) are repeatedly used each time the estimate of the image and the model parameters are updated. The iterative procedure is stopped when the difference of image intensity estimates between iterations becomes small. The

developments described from equation (4.22) through (4.49) constitutes the robust Kalman algorithm for the NSHP model (4.20) and the observation (4.24).

The above procedure is applied to restore realistic images. We depicted the same images as in Figure 3.3 for comparison. We implemented the robust Kalman algorithm along with the median filter, the robust algorithm developed in chapter 3, and the RUKF developed by Woods and Radewan [52].

4.6 Simulation and Results

The restoration algorithm based on the robust Kalman modelling approach is applied to the two test images shown in Figure 4.3. Both images are of size 512 x 512. Figure 4.3(a) is the original picture of a girl, and Figure 4.3(c) is the original picture of a baboon. The two images are digitized into 256 grey levels relative to 8 bits per pixel in resolution. We have chosen the same images as in chapter 3 to compare the effectiveness of the algorithms used in the simulation. A qualitative and quantitative comparison of the present algorithm to the median filter, the robust model developed in chapter 3, and the reduced update Kalman filter [52] was conducted.

To measure the performance of the different algorithms on noisy pictures, contaminated images are constructed by adding both Gaussian (0,100) noise and five percent of impulsive noise to the original images given in Figure 4.3(a) and (c). We used the same noise as in Figure 3.3 to compare the robust Kalman filter to the algorithm developed in the previous chapter. Figures 4.3(b) and 4.3(d) are the contaminated images of the girl and the baboon respectively. The impulse noise is generated to have equal probability of having grey level 0 (black) and 255 (white). In the simulation process, both ψ functions described in equations (3.14) and (3.15) were tested using different parameters and the results obtained show that the redescending ψ function (3.15) performed better than the hard-limiter type ψ function (3.14). This also confirms the superiority of the redescending ψ function over the

hard-limiter type ψ function. As we mentioned earlier, the new algorithm is an iterative robust Kalman filter which requires some knowledge about the observation noise statistics as well. After the first iteration, we assumed a white Gaussian distribution for the observation noise. Therefore, we can update the variance σ^2 , using equations (4.43)-(4.44). However, it is well known that, for a Gaussian distribution, more than 98% of the noise values falls within $\pm 3\sigma$. Therefore, we can choose the constants a , b , and c such that any outlier value is eliminated or reduced to fall within the Gaussian distribution limits. We choose $a=1.0$, $b=1.5$, and $c=2.5$ for the ψ function in Eq. (3.15) to update the observation noise.

In the simulation, we used different window sizes for the image data array. A window of size 8×8 performed the best. It is important to mention that the window size is critical in the algorithm proposed by Kashyap and his colleague. When a large window size is used, some of the more visible edges are distorted. However, in flat regions, the window size doesn't effect the performance of the algorithm. We used for convenience an 8×8 window as in chapter 3. The restored images of the girl, using the modified RUKF, the robust model, the median filter, and the reduced update Kalman filter are shown in Figure 4.4(a)-(d), respectively. Figure 4.5(a)-(d) are the restored images of the baboon following the same order. Comparing the visual quality of the restored images, Figure 4.4(a)-(d) for the girl, it can be seen that the noise is suppressed almost uniformly across the entire image, including the edges, for the new algorithm. There is little or no blurring around the edges contrary to the Kalman filtering techniques used. The RUKF performed well in flat regions, but there is still some impulse noise present in the restored image. The black spots, present in the image confirms that the conventional Kalman filter cannot be used for the restoration of images corrupted by impulse noise. The efficiency of the robust Kalman filter is further revealed in the restored baboon image, Figure 4.5(a). The restored image using the robust RUKF have better quality than those restored by the median filter, the reduced update Kalman filter, or the robust model proposed by Kashyap. The edges are sharp and even small details, such as the hair around the nose, are well preserved.

To measure the performance of the algorithms, several performance measures can be used such as the mean square-error (MSE), the absolute error (AE), and the signal-to-noise ratio (SNR). We use the mean square-error (MSE) between the originals and the restored images. A quantitative and qualitative comparison was conducted for all images. Figure 4.6 shows the individual error at each pixel along one row of the image data. Figure 4.6(a)-(b) show the error between the original images and their respective contaminated versions for both the girl picture and the baboon picture. Figure 4.6(c)-(d) represent the error between the original and the restored images using the modified reduced update Kalman filter for the girl and the baboon, respectively. Comparing the error in these images, it can be seen that the algorithm is very efficient in suppressing the noise in flat regions, where no edges are encountered, and performs very well around the edges, where little or no blurring is present.

Table 4.1 The Mean Square-Error between the original and the restored images, using different algorithms.

Image	MSE of robust RUKF	MSE of robust model	MSE of RUKF	MSE of median filter
Girl	21.462	33.313	194.798	51.072
Baboon	184.122	433.081	992.793	765.264

Table 4.1 shows the results, (MSE), obtained by the algorithms used. It can be seen that the MSE of the restored images, using the new approach, is lower than those obtained by the median filter, the robust model, or the reduced update Kalman filter. For the girl image, the MSE obtained by the modified RUKF, the RUKF, the robust model, and the median filter are 21.462, 194.798, 33.313, and 51.072,

respectively. The different algorithms performed well in suppressing the noise and the MSE values are much lower than the original MSE (1276.319). The mean square-error for the baboon picture, using the different algorithms, are also displayed in table 4.1. The results show the superiority of the new algorithm over the RUKF, the robust model, and the median filter. The new model developed had the best performance among all four techniques. The performance of the robust reduced update Kalman filter can be improved further, especially around edges. The local statistics can be introduced in the procedure to adapt better to significant changes in the image intensity (areas where an edge is present).

Figure 4.3 The original images and their noisy versions.
(a) The original image of the girl. (b) The original image of the baboon. (c) The corrupted image of the girl. (d) The corrupted image of the baboon.

a

b

c

d

2 of/de 2

MICROCOPY RESOLUTION TEST CHART
NATIONAL BUREAU OF STANDARDS
STANDARD REFERENCE MATERIAL 1010a
(ANSI and ISO TEST CHART No. 2)

Figure 4.4 The restored images of the girl by different algorithms.
(a) Restored image of the girl by the modified RUKF. (b) Restored image of the girl by the robust model. (c) Restored image of the girl by the median filter. (d) Restored image of the girl by the reduced update Kalman filter.

u

b

c

d

Figure 4.5 The restored images of the baboon using different algorithms.
(a) Restored image of the baboon by the modified RUKF. (b) Restored image of the baboon by the robust model. (c) Restored image of the baboon by the median filter.
(d) Restored image of the baboon by the reduced update Kalman filter.

a

b

c

d

Figure 4.6 Error measurement along line (30,j).

(a) Error between the original and the corrupted image of the girl. (b) Error between the original and the corrupted image of the baboon. (c) Error between the original girl image and the restored girl image using the modified RUKF. (d) Error between the original baboon image and the restored baboon image using the modified RUKF.

Chapter 5

SUMMARY AND CONCLUSIONS

In one sense, the aim of computer vision is to extract and interpret the organization of images in ways which are useful in specific applications. However, images acquired in realistic environments inevitably suffer from various types of degradations, including noise, blurring, geometrical distortion etc.. Images are effected by noise from a variety of different sources, such as sensor noise, quantization noise, grain noise, and channel noise. The noise may be systematic or random, the former being the easier to deal with in practice. Images could also be degraded by blur which may be due to the relative motion between the sensor and the object, atmospheric disturbances, etc.. These degradations pose many problems for implementing computer vision techniques and often lead to undesirable results. However, it is necessary to consider ways by which the quality of images could be improved and techniques that reduce the effect of such degradations. Image restoration is certainly the most appropriate technique to improve the quality of images for the human viewer as well as for man machine interaction in automation and artificial intelligence.

A wide variety of image restoration techniques have been developed during the last two decades or so. An extensive literature review of image restoration techniques available has been conducted, and the interesting areas requiring further studies have been identified. Of these areas, nonstationary image restoration, robust modelling, and robust restoration are pursued extensively in this thesis research. It has been shown that additive noise and motion blur are the most common forms of degradations which have been investigated by researchers. In most of the restoration algorithms, the noise is assumed to be additive, signal-independent, white, and Gaussian. These assumptions are made for the convenience of mathematical derivation and computational efficiency. They sometimes lead to results that are far from being realistic. Images are often assumed to be wide-sense stationary, and the image field is modeled as a linear shift-invariant system (LSI). This is not the case for most images as the image intensity could change abruptly in the presence of edges. The algorithms using the stationarity assumption tend to smooth out the noise as well as any abrupt changes in the image intensity. As a result, the restored images are smooth but blurred.

In the framework of stochastic estimation theory, the one-dimensional Kalman filter is considered as the best linear recursive estimation technique. It has been extended to two dimensional applications with some degree of success. A sequential Kalman filter for the restoration of images corrupted by additive Gaussian noise have been developed. A spatial causal image field is defined which divides the entire image into three regions, namely, the "Past", the "Present", and the "Future". Based on this partitioning of the image field, a linear state equation model is proposed. The two-dimensional image intensity array is ordered into a long string of pixels following a square spiral scanning operation rather than the conventional raster scan. With this scanning operation, the Kalman filter behaves like a one-dimensional estimator by processing one pixel at a time. The Kalman equations are obtained based on the stationarity assumption. The minimum variance optimality criterion have been used on the global image field. However, the minimum variance criterion result in algorithms unable to respond to abrupt changes in the image intensity which often

result in smooth but blurred images. In order to handle this problem, the stationarity assumption is used locally and a weighting function which varies with the local spatial activity is proposed. The weighting function is applied to the Kalman gains by using the local spatial statistics measurements. While the underlying idea of this approach is similar to some previously developed algorithms, the salient feature is that the sequential Kalman filter is derived based on a stationary image model. This makes the derivation straightforward and simple. The actual nonstationarity of the image is handled by local statistics measurement. Consequently, the sequential Kalman filter is able to adjust to the local characteristics of the image. The simulation results show that the sequential Kalman filter is able to preserve the edges while smoothing out the noise. The quality of the images tested is significantly improved.

The effectiveness of a restoration method depends mostly on how well the image model fits the image data. The image intensity array is usually assumed to be a multivariate Gaussian distribution. The Gaussian assumption is used primarily to estimate the parameters of the model. The least squares method works perfectly under the Gaussian noise assumption, but when the image contains outliers the least squares fails to estimate the parameters properly, and a robust estimator is needed to overcome this problem. Even though a robust procedure is necessary in most of the image processing applications, very little research has been done on the use of robust procedures in image restoration. We developed estimation algorithms for the nonsymmetric half-plane autoregressive image model and apply these robust methods to the restoration of images contaminated by impulsive noise.

We implement a robust procedure proposed by Kashyap and Eom [27]. The robust estimator is a maximum likelihood estimator which minimizes a nonquadratic cost function with several arguments rather than a quadratic function in the case of least squares. An iterative estimation algorithm is presented in which the driving noise is a mixture of a Gaussian and outliers. We consider the case where the original image is not available and a noisy version of the image is available. The robust procedure estimates the model parameters and the original image intensity simultaneously, assuming that the entire image obeys a nonsymmetric half-plane

model. The simulation is conducted by applying the robust estimation procedure to realistic images corrupted by a mixture type noise. The results obtained confirm the superiority of such technique over some of the classical methods used to treat impulsive noise.

The Kalman filter is not effective when the noise is impulsive. We consider robust parameter estimation and Kalman filtering simultaneously to restore images contaminated by a mixture of Gaussian noise and impulse noise. For reasons of computational and storage efficiency, the reduced update Kalman filter is used instead of the sequential Kalman filter developed earlier. Further gains in terms of computations and storage requirement are obtained by dividing the image field into small windows in which the image is assumed to obey an NSHP autoregressive model. The underlying procedure is a combination of robust parameter estimation and a modified version of the reduced update Kalman filter. The resulting Kalman filter is applied to restore realistic images containing a small percentage of outliers. The simulation results show that the Kalman filter is robust in the sense that it removes impulse noise effectively. The Gaussian portion of the noise is also removed and is hardly noticeable in the restored images. The salient characteristic of the modified reduced update Kalman filter is its ability to preserve edges after restoration without causing the artifacts often pertinent in Kalman filtering. Qualitative and quantitative evaluations have been conducted and the results obtained from the robust Kalman filter compared favorably to the reduced update Kalman filter, the median filter, and the robust estimation procedure proposed.

While most of the model based restoration techniques are oriented towards modelling the image field, little attention has been devoted to the observation model. Many assumptions about the noise distribution and the image intensity distribution are imposed before the development of restoration procedures. These assumptions, whether they are justified or not, are used for the convenience of mathematical derivation and computational efficiency. For example, if the image is assumed to be stationary, the statistical properties of the image are governed globally rather than locally by its stationary covariance matrix which is of Toeplitz form. under this

assumption, the necessary computation can be carried out by FFT algorithms and consequently, the computation time is reduced drastically. The savings in computation time are, however, at the expense of restoration quality. It has been shown that most images are nonstationary and modelling the image by a wide-sense stationary field often lead to blurred images. However, nonstationary image restoration appears to be one of the possible solutions to the smoothing problem and to the artifacts usually present around the edges.

As we mentioned earlier, little effort has been devoted to the observation model. Modeling the observation and the noise more properly could result in restoration procedures that effectively reduce the effect of noise without distorting the structure of the image. The local statistical information could be incorporated into the observation model where the noise could be modeled appropriately. The nonlinearity of received image signals is also a problem that is not well investigated, and more research have to be done in the area of nonlinear image restoration.

More research is certainly needed in the area of robust modeling, and stochastic estimation techniques are the most appropriate methods that could be extended to robust procedures for image processing tasks. Further research might be necessary in the area of evaluating other sources of noise. For instance, the sensor noise is usually ignored in image models. However, it is possible that the effect of such noise is significant in the development of such models.

In most cases, the entire corrupted image is available. Thus, the whole image intensity array is available. So, instead of modelling the image by an nonsymmetric half plane autoregressive model, the image could be modelled by a noncausal model. In this case, the image model is more likely to fit the image properly. The stationarity problem could be solved more easily. The close neighbourhood of a pixel will consist of at least eight neighbours which have the most significant correlations. As a recommendation for further research, iterative procedures are more likely to be used in noncausal models. The robust modelling procedures developed in this thesis could be extended to noncausal image modelling.

Bibliography

- [1] Abramatic J. F. and L. M. Silverman (1982). Nonlinear restoration of noisy images. *IEEE Trans. Pattern Recog. Mach. Intell.*, PAMI-4, No. 2, pp. 141-149.
- [2] Alpach D. L. (1974). A parallel filtering algorithm for linear systems with unknown time varying noise statistics, *IEEE Trans. Automat. Contr.*, pp. 552-560.
- [3] Anderson B. D. O. and J. B. Moore (1972). The Kalman-Bucy filter as a true time-varying Wiener filter. *IEEE Trans. Syst., Man, Cybern.*, SMC-1, No. 2, pp. 119-128.
- [4] Anderson G. L. and A. N. Netravali (1976). Image restoration based on a subjective criterion. *IEEE Trans. Syst., Man, Cybern.*, SMC-6, No. 6, pp. 845-853.
- [5] Andrews H. C. (1974). Digital image processing: a survey. *Computer*, Vol. 7, No. 5, pp. 34-45.
- [6] Andrews H. C. and B. R. Hunt (1977). *Digital image restoration*. Prentice-Hall, N.J..
- [7] Biemond J., J. Rieske, and J. J. Gerbrands (1983). A Fast Kalman filter for image degraded by both blur and noise. *IEEE Trans. Acoust., Speech, Signal Processing*, ASSP-31, No.5, pp. 1248-1255.
- [8] Bryson A. E., Jr. and Y. C. Ho (1969). *Applied optimal control*, Waltham, MA: Ginn and Co.
- [9] Bovik A. C. et Al. (1985). Edge-sensitive image restoration using order constrained least-squares method, *IEEE Trans. Acoust., Speech, Signal Processing*, ASSP-33, No. 4, pp. 1253-1263.

- [10] Chan P. and J. S. Lim (1985). One-dimensional processing for adaptive image restoration, *IEEE Trans. Acoust., Speech, Signal Processing*, ASSP-33, No. 1, pp. 117-125.
- [11] Deutsch R. (1965). *Estimation Theory*, Prentice-Hall Inc., Englewood Cliffs, N. J., pp. 206-229.
- [12] Franks L.E. (1966). A model for the random video process. *Bell Syst. Tech. J.*, Vol. 45, pp. 609-630.
- [13] Godbole S. S. (1974). Kalman filtering with no a priori information about noise-White noise case: Identification of covariances, *IEEE Trans. Automat. Contr.*, No. 10, pp. 561-563.
- [14] Gonzalez R. C. and P. Wintz (1984). *Digital Image Processing*, Second Edition, Addison-Wesley.
- [15] Goodwin G. C. and K. S. Sin, (1984), *Adaptive Filtering, Prediction and Control*. Information and system science series, Prentice-Hall, New Jersey.
- [16] Green W. B. (1983). *Digital image processing: A systems approach*. Van Nostrand Reinhold company, New York.
- [17] Habibi A. (1972). Two-dimensional Bayesian estimation of images. *Proc. IEEE*, Vol. 60, pp. 878-883.
- [18] Henry Stark (1987). *Image Recovery: Theory and application*, Academic Press.
- [19] Ho Y. C. (1972). On the minimax principle and zero-sum stochastic differential games, Tech. Rep. No. 630, Div. Engineering and Appl. Physics, Harvard University, Cambridge, MA.
- [20] Hunt B. R (1977). Bayesian methods in nonlinear digital image restoration. *IEEE Trans. Comput.*, C-26, No. 3, pp. 219-229.
- [21] Hunt B. R. and T. M. Cannon (1976). Nonstationary assumptions for Gaussian models of images, *IEEE Trans. Syst., Man, Cybern.*, SMC-6, No. 12, pp. 876-881.
- [22] Ingle V. K., A. Radpour, J. W. Woods, and H. Kaufman. Recursive estimation with non-homogeneous image models, *Proc. IEEE Pattern Recog. Image Processing*. Chicago, IL, pp. 105-108.
- [23] Jain A. K. (1977). A semicausal model for recursive filtering of two-dimensional

- images, IEEE Trans. Comput., C-26, No. 4, pp. 343-350.
- [24] Jain A. K. and E. Angel (1974). Image restoration, modelling, and reduction of dimensionality, IEEE Trans. Comput., C-23, No. 5, pp. 470-476.
- [25] Jeng F. -C. and J. W. Woods (1988). Inhomogeneous Gaussian image models for estimation and restoration, IEEE Trans. Acoust., Speech, Signal Processing, ASSP-36, No. 8, pp. 1305-1312.
- [26] Kalman R. E. (1960). A new approach to linear filtering and prediction problems. J. Basic Engineering, (March, ASME Trans.), pp. 35-45.
- [27] Kashyap R. L. and K. -B. Eom (1988). Robust image modelling techniques with image restoration application, IEEE Trans. Acoust., Speech, Signal Processing, ASSP-36, No. 8, pp. 1313-1325.
- [28] Kaufman H., J. W. Woods, S. Dravida, and A. M. Tekalp (1983). Estimation and identification of two-dimensional images. IEEE Trans. Automat. Contr., AC-28, No. 7, pp. 745-756.
- [29] Kondo H. et al. (1981). Improved methods of Bayesian estimation for nonlinear image restoration. Trans. Inst. Electron. Commun., Eng. Jnp. Sect, E(Engl.). V. E64, No. 11, pp. 744-745.
- [30] Kuan D. T., A. A. Sawchuk, T. C. Strand, and P. Chavel (1984). Nonstationary 2-D recursive restoration of images with signal-dependent noise. Proc. Int. Conf. Acoust., Speech, Signal Processing, ICASSP-84, San Diego, CA., 37.4.1-37.4.4.
- [31] Lee J. -S. (1980). Digital image enhancement and noise filtering by use of local statistics. IEEE Trans. Pattern Analy., Machine Intel., PAMI-2, No. 2, pp. 165-168.
- [32] Lee J. -S. (1981). Refined filtering of image noise using local statistics. Computer Graph. and image processing, Vol.15, pp. 380-389.
- [33] Ljung L. and T. Söderström (1983). Theory and practice of recursive identification. MIT press, Cambridge, MA.
- [34] Morris J. M. (1976). The Kalman filter: a robust estimator for some classes of linear quadratic problems, IEEE Trans. Info. Theory, Vol. IT-22, No. 5, pp. 526-534.

- [35] Murphy M. S. and L. M. Silverman (1978). Image model representation and line by-line recursive restoration, *IEEE Trans. Automat. Cont.*, AC-23, No. 5, pp. 809-816.
- [36] Nahi N. E. (1969). *Estimation Theory and Application*. New York, NY: Wiley.
- [37] Nahi N. E. and T. Assafi (1972). Bayesian recursive image estimation, *IEEE Trans. Comput.* (short notes), Vol. C-12, pp.734-738.
- [38] Nahi N. E. and A. Habibi (1975). Decision-directed image enhancement, *IEEE Trans. Circui. Syst.*, CSA-22, No. 3, pp. 286-293.
- [39] Netravali A. N. and B. Brasada (1975). Adaptive quantization of picture signals based on spatial masking, *Proc. IEEE*, Vol. 65, No. 4, pp. 536-548.
- [40] Papoulis A. (1984). *Probability, Random Variables, and Stochastic Processes*. 2nd edition, McGraw-Hill Book Company, New York.
- [41] Powell S. R. and L. M. Silverman (1974). Modeling of two dimensional covariance functions with application to image restoration. *IEEE Trans. Automat. Contr.*, AC-19, No. 1, pp. 8-13.
- [42] Rajala S. A. and R. P. DeFigueiredo (1981). Adaptive nonlinear image restoration by a modified Kalman filtering approach, *IEEE Trans. Acoust., Speech, Signal Processing*, ASSP-29, No. 5, pp. 1033-1042.
- [43] Rosenfeld A. and A. C. Kak (1982). *Digital picture processing*, Academic Press, 2nd Edition, New York.
- [44] Sage A. P. and J. L. Melsa (1971). *Estimation Theory*. McGraw-Hill Book Company, New York.
- [45] Schafer R. W. and R. M. Mersereau (1981). Constrained iterative restoration algorithms, *Proc. IEEE*, Vol. 69, pp. 432-450.
- [46] Strickland R. N. (1985). Estimation of local statistics for digital processing of nonstationary images, *IEEE Trans. Acoust., Speech, Signal Processing*, ASSP-33, No. 2, pp. 465-469.
- [47] Tekalp A. M., H. Kaufman, and J. W. Woods (1989). Edge-Adaptive Kalman Filtering for Image Restoration with Ringing Suppression, *IEEE Trans. Acoust., Speech, Signal Processing*, ASSP-37, No. 6, pp. 892-898.

- [48] Trussell H. J. and B. R. Hunt (1978). Sectioned methods for image restoration. IEEE Trans. Acoust., Speech, and Signal Processing, ASSP-26, No. 2, pp. 157-164.
- [49] Trussell H. J. and B. R. Hunt (1979). Improved methods of maximum a posteriori image restoration. IEEE Trans. Comput., C-28, NO. 1, pp. 57-62.
- [50] Wallis R. (1976). An approach to the space-variant restoration and enhancement of images. Proc. Symp. Current Math. Problems in Image Science, Naval Postgrad. School, Monterey, CA., Ed. C. O. Wilde and E. Barrett, pp. 107-111.
- [51] Woods J. W. and C. H. Radewan (1976). Reduced update Kalman filter: A two dimensional recursive processor, John Hopkins Conf. Inform. Sci. Syst., Baltimore, MD..
- [52] Woods J. W. and C. H. Radewan (1977). Kalman filtering in two dimensions, IEEE Trans. Info. Theory, IT-23, No. 4, pp. 473-482.
- [53] Woods J. W. (1979). Correction to "Kalman filtering in two dimensions" IEEE Trans. Infor. Theory, IT-25, No. 5, pp. 628-629.
- [54] Woods J. W. and V. K. Ingle (1981). Kalman filtering in two dimensions: Further results, IEEE Trans. Acoust., Speech, Signal Processing, ASSP-29, No. 2, pp. 188-197.
- [55] Wu Z. (1985). Multidimensional state-space model Kalman filtering with application to image restoration, IEEE Trans. Acoust., Speech, Signal Processing, ASSP-33, No. 6, pp. 1576-1592.

Appendices

Appendix A: Definitions

Definition of a Random Field

We consider a family of functions $f(p, w_i)$ over the set of all outcomes $w_i = \{w_1, w_2, \dots\}$ of an experiment, where $p \in P$, and P is an interval on the real axis or a region of a multidimensional Euclidean space. When P is one-dimensional, the family of functions $f(p, w_i)$ is called a stochastic process. When the dimensionality of P is greater than one, the family of functions $f(p, w_i)$ is defined as a *random field*.

In image processing, images are considered as two-dimensional random fields. In the sequel, we shall discuss some properties of random fields with P being two-dimensional.

Let P define the XY -plane, so that p is a point in the XY -plane, and p can be represented by its coordinates, i.e., $p = (x, y)$. With this notation, the random field $f(p, w_i)$ can be expressed as $f((x, y), w_i)$. For a given value of (x, y) , $f((x, y), w_i)$ is a random variable, while for a given outcome w_i , $f((x, y), w_i)$ is a function over the XY -plane.

Mean, Correlation and Covariance

It is known that, in general, the density function of a random variable depends on the value of (x,y) chosen. As a consequence, its expected value must also be a function of (x,y) . Let $m(x,y)$ denote this expectation, thus

$$m(x,y) = E[f(x,y)] = \int_1 \int_2 p(t_1, t_2; x, y) dt_1 dt_2$$

The correlation of two random fields $f(x,y)$ and $g(x,y)$ is defined as the expected value of the product of the two random variables $f(x_1, y_1)$ and $g(x_2, y_2)$:

$$\begin{aligned} R_{fg}(x_1, y_1; x_2, y_2) &= E[f(x_1, y_1) g(x_2, y_2)] \\ &= \int_1 \int_2 t_1 t_2 p(t_1, t_2; x_1, y_1, x_2, y_2) dt_1 dt_2 \end{aligned}$$

The covariance of two random fields is defined as

$$\begin{aligned} \text{Cov}_{fg}(x_1, y_1; x_2, y_2) &= \\ &E\{[f(x_1, y_1) - m_f(x_1, y_1)] [g(x_2, y_2) - m_g(x_2, y_2)]\} \\ &= R_{fg}(x_1, y_1; x_2, y_2) - m_f(x_1, y_1) m_g(x_2, y_2). \end{aligned}$$

If $\text{Cov}_{fg}(x_1, y_1; x_2, y_2) = 0$ for all (x_1, y_1) and (x_2, y_2) , the two random fields $f(x_1, y_1)$ and $g(x_2, y_2)$ are said to be uncorrelated. This is equivalent to

$$E[f(x_1, y_1) g(x_2, y_2)] = E[f(x_1, y_1)] E[g(x_2, y_2)]$$

Wide Sense Stationary Random Fields

The concept of two-dimensional stationary random fields is an extension of one-dimensional wide sense stationary stochastic process. A random field is called stationary if its expected value $m(x,y)$ is independent of position (x,y) in the XY-plane, i.e.,

$$m(x,y) = m = \text{constant.}$$

and if its autocorrelation function is shift invariant,

$$R_{ff}(x_1, y_1; x_2, y_2) = E[f(x_1 - x_2, y_1 - y_2) f(x_1 - x_2, y_1 - y_2)]$$

for all (x_1, y_1) and (x_2, y_2) in the XY-plane, the random field is wide-sense stationary. By setting $i = x_1 - x_2$, and $j = y_1 - y_2$, the equation above can be rearranged as

$$R_{ff}(x_1, y_1; x_2, y_2) = R_{ff}(x_1 - x_2, y_1 - y_2) = R_{ff}(i, j).$$

Similarly, two random fields $f(x,y)$ and $g(x,y)$ are jointly stationary, if

$$R_{fg}(x_1, y_1; x_2, y_2) = R_{fg}(i, j).$$

Appendix B: Listings of Simulation Programs

```
/******
```

NOISE.C

This program is designed to generate Gaussian noise. Part of this program generates the outlier sequence such that the Gaussian noise could be contaminated by impulsive noise.

```
*****/
```

```
#include <stdio.h>
#include <stdlib.h>
#include <math.h>
#include <pipdrv.h>
```

```
#define M      512
#define N      416

#define M1     259200
#define IA1     7141
#define IC1     54773
#define RM1     (1.0/M1)
#define M2     134456
#define IA2     8121
#define IC2     28411
#define RM2     (1.0/M2)
#define M3     243000
#define IA3     4561
#define IC3     51349
```

```
float ran1(idum)
int *idum;
{
    static long ix1,ix2,ix3;
    static float r[98];
    float temp;
    static int iff=0;
    int j;
    if(*idum<0 || iff==0){
        iff=1;
        ix1=(IC1-( *idum)) % M1;
        ix1=(IA1*ix1+IC1) % M1;
        ix2=ix1 % M2;
```

```

ix1=(IA1*ix1+IC1) % M1;
ix3=ix1 % M3;
for(j=0;j<97;j++){
    ix1=(IA1*ix1+IC1) % M1;
    ix2=(IA2*ix2+IC2) % M2;
    r[j]=(ix1+ix2*RM2)*RM1;
}
*idum=1;
}
ix1=(IA1*ix1+IC1) % M1;
ix2=(IA2*ix2+IC2) % M2;
ix3=(IA3*ix3+IC3) % M3;
j=1+((96*ix3)/M3);
if(j>96 || j<0) printf("RAN1: This can't happen..");
temp=r[j];
r[j]=(ix1+ix2*RM2)*RM1;
return temp;
} /* RAN1 */

/* Generating a Gaussian distribution from the random numbers */

float gasdev(idum)
int *idum;
{
    static int iset=0;
    static float gset;
    float fac,r,v1,v2;
    float ran2();
    if(iset==0){
        do{
            v1=2.0*ran2(idum)-1.0;
            v2=2.0*ran2(idum)-1.0;
            r=v1*v1 + v2*v2;
        } while(r>=1.0);
        fac=sqrt(-2.0*log(r)/r);
        gset=v1*fac;
        iset=1;
        return v2*fac;
    } else {
        iset=0;
        return gset;
    }
} /* gasdev */

#define MM 714025
#define IA 1366
#define IC 150889

float ran2(idum)
long *idum;

```



```

{
    static long iy,ir[98];
    static int iff=0;
    int j;
    if(*idum<0 || iff ==0) {
        iff=1;
        if(((*idum=(1C-(*idum)) % MM) <0) *idum = -(*idum);
        for(j=1;j<=97;j++) {
            *idum=(1A*( *idum)+1C) % MM;
            ir[j]=(*idum);
        }
        *idum=(1A*( *idum)+1C) % MM;
        iy=(*idum);
    }
    j=1+97.0*iy/MM;
    if(j>97 || j<1) printf("RAN2: this can't happen");
    iy=ir[j];
    *idum=(1A*( *idum)+1C) % MM;
    ir[j]=(*idum);
    return (float) iy/MM;
} /* RAN2 */

```

```

main()
{
    float std,Mean,Var,
        noise,index,outlier;
    int x,y,val,idum=(-11);
    long int pos;
    Mean =0.0;Var=0.0;

    /* Corrupting the image by a Gaussian noise N(0,std) */

    printf("Enter the standard deviation std:");
    scanf(" %f",&std);
    for( y=1; y < N; y++)
    {
        printf("\rY= %d",y);
        for( x=1; x< M-1; x++)
        {
            pos=y*512L+x;
            val=pixel(x,y);
            noise = std*gasdev(idum);
            index = val + noise;
            /* if((pos % 10)==0){
                outlier=ran1(-13);
                if(outlier<0.5) index=0.0;
                else index=255.0;
            }*/
            if(index<0.0) index=0.0;
            if(index>255.0) index=255.0;

```

```

    plot(x,y,(int)index);

    Mean += noise/(M-2.0);
    Var += noise*noise/(M-2.0);
    } /* for */
} /* for */
Mean = Mean/(N-1.0);
Var = (Var/(N-1.0)) - (Mean*Mean);
printf("\n\n Mean = %.5f  Var = %.3f \n\n",Mean,Var);

} /* main */

```

```

;*****

```

SPIRAL.C

This is a sequential Kalman Filter routine for the restoration of images corrupted by white Gaussian noise. The modeling coefficients and the covariance of the modeling error were calculated using the least-squares method. A spiral scanning operation is used for this particular routine.

```

*****/

```

```

#include <stdio.h>
#include <math.h>
#include <pipdrv.h>

```

```

#define NC 511
#define NR 416
#define D 4
#define VN 400.0

```

```

static float sqrarg;
#define SQR(a) (sqrarg=(a),sqrarg*sqrarg)

```

```

static float Vfd0[D][D]=
    {1497.39,1480.41,1472.05,1433.34,
     1480.41,1497.39,1476.78,1435.35,
     1472.05,1476.78,1497.39,1469.52,
     1433.34,1435.35,1469.52,1497.39};

```

```

float A[D]= {0.601887,-0.39702,0.679901,0.1180321};

```

```

main()
{

    int x,y,i,j,k,l,
        inc1,inc2;
    float K,Varn,chisq,Vfd[D][D],VR[NC][4],
        VD[NR][4],VL[NC][4],VU[NR][4];

    chisq=10.04;
    Varn=VN;

    for(k=0;k<NC;k++){
        for(j=0;j<D;j++){
            VR[k][j]=Vfd0[0][0];
            VL[k][j]=Vfd0[0][0];
        }
        for(j=0;j<NR;j++){
            for(i=0;i<D;i++){
                VD[j][i]=Vfd0[0][0];
                VU[j][i]=Vfd0[0][0];
            }
        }
        for(i=0;i<D;i++){
            for(j=0;j<D;j++) Vfd[i][j]=Vfd0[i][j];
        }

        for(inc1=1,inc2=1;(inc1<255 && inc2<208);inc1++,inc2++){
            y=inc2;
            printf("G%d\r",y);
            right_scan(y,inc1,Varn,chisq,Vfd,VR,A);
            x=NC-1-inc1;
            down_scan(x,inc2,Varn,chisq,Vfd,VD,A);
            y=NR-1-inc2;
            left_scan(y,inc1,Varn,chisq,Vfd,VL,A);
            x=inc1;
            up_scan(x,inc2,Varn,chisq,Vfd,VU,A);
        }
    } /* main */

    local_mean(x,y)
    int x,y;
    {
        int i,j;
        float m=0.0;
        for(i=0;i<3;i++){
            for(j=0;j<3;j++) m += (float) pixel(x+j-1,y+i-1);
        }
        m /=9.0;
        return m;
    }

```

```

local_var(x,y,m)
int x,y;
float m;
{
int i,j;
float var=0.0;
for(i=0;i<3;i++)
    for(j=0;j<3;j++) var += SQR(pixel(x+j-1,y+i-1)-m);
var /=9.0;
return var;
}

mean(x,y)
int x,y;
{
int i,j;
float m=0.0;
for(i=0;i<3;i++)
    for(j=0;j<3;j++) m += (float) pixel(x+j-1,y+i-1);
m /=9.0;
return m;
}

var(x,y,m)
int x,y;
float m;
{
int i,j;
float var=0.0;
for(i=0;i<3;i++)
    for(j=0;j<3;j++) var += SQR((float) pixel(x+j-1,y+i-1)-m);
var /=9.0;
return var;
}

right_scan(y,inc1,Varn,ws,Vfd,VR,A)
int y,inc1;
float Varn,ws,Vfd[4][4],A[4],VR[NC][4];
{
int x,i,j,k,afunc[D];
float fhat,fhm,ennov,KG,
    m11,m12,m13,m21,m22,m23,m31,m32,m33,
    d1,d2,d3,d4,lvar,m,
    Vfdmm[D][D],Vfdm[D][D];
for(x=inc1;x<=(NC-1-inc1);x++){
    afunc[0]=pixel(x-1,y);afunc[1]=pixel(x-1,y-1);
    afunc[2]=pixel(x,y-1);afunc[3]=pixel(x+1,y-1);
    fhm=0.0;
    for(j=0;j<D;j++) fhm += A[j]*afunc[j];
    ennov=(float) pixel(x,y)-fhm;
}
}

```

```

if(y >= 4 && x >= 4 && x < NC-4){
    m=mean(x,y);
    lvar=var(x,y,m);
    lvar -= VN;
    if(lvar > VN) Varn = VN/2.0;
    else Varn=VN;
    if(lvar > VN+50.0){
        m11=local_mean(x-3,y-3);
        m12=local_mean(x,y-3);
        m13=local_mean(x+3,y-3);
        m21=local_mean(x-3,y);
        m22=local_mean(x,y);
        m23=local_mean(x+3,y);
        m31=local_mean(x-3,y+3);
        m32=local_mean(x,y+3);
        m33=local_mean(x+3,y+3);
        d1=fabs(m11-m33)+fabs(m12-m23)+fabs(m21-m32);
        d2=fabs(m11-m31)+fabs(m12-m32)+fabs(m13-m33);
        d3=fabs(m12-m21)+fabs(m13-m31)+fabs(m23-m32);
        d4=fabs(m13-m11)+fabs(m23-m21)+fabs(m33-m31);

        if(d1 > d2 && d1 > d3 && d1 > d4){
            if(fabs(m11-m22) <= fabs(m22-m33)) lvar=local_var(x-3,y-3,m11);
            else lvar=local_var(x+3,y+3,m33);
        }
        if(d2 > d1 && d2 > d3 && d2 > d4){
            if(fabs(m12-m22) <= fabs(m22-m32)) lvar=local_var(x,y-3,m12);
            else lvar=local_var(x,y+3,m32);
        }
        if(d3 > d1 && d3 > d2 && d3 > d4){
            if(fabs(m13-m22) <= fabs(m22-m31)) lvar=local_var(x+3,y-3,m13);
            else lvar=local_var(x-3,y+3,m31);
        }
        if(d4 > d1 && d4 > d2 && d4 > d3){
            if(fabs(m23-m22) <= fabs(m22-m21)) lvar=local_var(x,y+3,m23);
            else lvar=local_var(x,y-3,m21);
        }
        lvar -= VN;
        if(lvar > 0.0) Varn = VN*exp(-200.0*lvar);
        else Varn=VN;
    }
}

if((y > 1) && (x >= 1) && (x < NC-1))
    for(i=1; i < D; i++){
        if(i==1)
            for(j=i; j < D; j++) Vfd[i][j]=Vfd[j][i]=VR[x-1][j-i];
        if(i==2)
            for(j=i; j < D; j++) Vfd[i][j]=Vfd[j][i]=VR[x][j-i];
        if(i==3)
            for(j=i; j < D; j++) Vfd[i][j]=Vfd[j][i]=VR[x+1][j-i];
    }

```

```

    }

    for(i=0;i<D;i++){
        for(j=0;j<D;j++){
            if(i==0){
                Vfdmm[0][j]=0.0;
                for(k=0;k<D;k++) Vfdmm[i][j] += A[k]*Vfd[k][j];
            }
            else Vfdmm[i][j]=Vfd[i][j];
        }
    }
    for(i=0;i<D;i++){
        for(j=0;j<D;j++){
            if(j==0){
                Vfdm[i][j]=0.0;
                for(k=0;k<D;k++) Vfdm[i][j] += Vfdmm[i][k]*A[k];
            }
            else Vfdm[i][j]=Vfd[i][j];
        }
    }
    Vfdm[0][0] +=ws;
    KG = Vfdm[0][0]/(Vfdm[0][0]+Varn);
    fhat=fhm + KG*ennov;
    if(fhat<=0.0) fhat=0.0;
    if(fhat>=255.0) fhat=255.0;
    plot(x,y,(int) fhat);
    for(i=0;i<D;i++){
        Vfd[0][i]=Vfd[i][0]=(1-KG)*Vfdm[0][i];
        VR[x][i]=Vfd[0][i];
    }
    Varn=VN;
}

} /* right_scan */

```

```

down_scan(x,inc2,Varn,ws,Vfd,VD,A)
int x,inc2;
float Varn,ws,Vfd[4][4],A[4],VD[NR][4];
{
    int y,i,j,k,afunc[D];
    float fhat,fhm,ennov,rn,KG,
        m11,m12,m13,m21,m22,m23,m31,m32,m33,
        d1,d2,d3,d4,lvar,
        Vfdmm[D][D],Vfdm[D][D];

    for(y=inc2;y<(NR-1-inc2);y++){
        afunc[0]=pixel(x,y-1);afunc[1]=pixel(x+1,y-1);
        afunc[2]=pixel(x+1,y);afunc[3]=pixel(x+1,y+1);
        fhm=0.0;
        for(j=0;j<D;j++) fhm += A[j]*afunc[j];
        ennov=(float) pixel(x,y)-fhm;
    }
}

```

```

if(y>=4 && x<NC-4 && y<NR-4){
    m=mean(x,y);
    lvar=var(x,y,m);
    lvar -=VN;
    if(lvar>VN) Varn = VN/2.0;
    else Varn=VN;
    if(lvar>VN+50.0){
        m11=local_mean(x-3,y-3);
        m12=local_mean(x,y-3);
        m13=local_mean(x+3,y-3);
        m21=local_mean(x-3,y);
        m22=local_mean(x,y);
        m23=local_mean(x+3,y);
        m31=local_mean(x-3,y+3);
        m32=local_mean(x,y+3);
        m33=local_mean(x+3,y+3);
        d1=fabs(m11-m33)+fabs(m12-m23)+fabs(m21-m32);
        d2=fabs(m11-m31)+fabs(m12-m32)+fabs(m13-m33);
        d3=fabs(m12-m21)+fabs(m13-m31)+fabs(m23-m32);
        d4=fabs(m13-m11)+fabs(m23-m21)+fabs(m33-m31);

        if(d1>d2 && d1>d3 && d1>d4){
            if(fabs(m11-m22)<=fabs(m22-m33)) lvar=local_var(x-3,y-3,m11);
            else lvar=local_var(x+3,y+3,m33);
        }
        if(d2>d1 && d2>d3 && d2>d4){
            if(fabs(m12-m22)<=fabs(m22-m32)) lvar=local_var(x,y-3,m12);
            else lvar=local_var(x,y+3,m32);
        }
        if(d3>d1 && d3>d2 && d3>d4){
            if(fabs(m13-m22)<=fabs(m22-m31)) lvar=local_var(x+3,y-3,m13);
            else lvar=local_var(x-3,y+3,m31);
        }
        if(d4>d1 && d4>d2 && d4>d3){
            if(fabs(m23-m22)<=fabs(m22-m21)) lvar=local_var(x,y+3,m23);
            else lvar=local_var(x,y-3,m21);
        }
        lvar -=VN;
        if(lvar>0.0) Varn = VN*exp(-200.0*lvar);
        else Varn=VN;
    }
}

```

```

if((y>1) && (y<NR-1) && (x<NC-1))
    for(i=1;i<D;i++){
        if(i==1)
            for(j=i;j<D;j++) Vfd[i][j]=Vfd[j][i]=VD[y-1][j-i];
        if(i==2)
            for(j=i;j<D;j++) Vfd[i][j]=Vfd[j][i]=VD[y][j-i];
        if(i==3)

```

```

        for(j=i;j<D;j++) Vfd[i][j]=Vfd[j][i]=VD[y+1][j-i];
    }

    for(i=0;i<D;i++){
        for(j=0;j<D;j++){
            if(i==0){
                Vfdmm[0][j]=0.0;
                for(k=0;k<D;k++) Vfdmm[0][j] += A[k]*Vfd[k][j];
            }
            else Vfdmm[i][j]=Vfd[i][j];
        }
    }
    for(i=0;i<D;i++){
        for(j=0;j<D;j++){
            if(j==0){
                Vfdm[i][j]=0.0;
                for(k=0;k<D;k++) Vfdm[i][j] += Vfdmm[i][k]*A[k];
            }
            else Vfdm[i][j]=Vfd[i][j];
        }
    }
    Vfdm[0][0] += ws;
    KG = Vfdm[0][0]/(Vfdm[0][0]+Varn);
    fhat=fhm + KG*ennov;
    if(fhat<=0.0) fhat=0.0;
    if(fhat>=255.0) fhat=255.0;
    plot(x,y,(int) fhat);
    for(i=0;i<D;i++){
        Vfd[0][i]=Vfd[i][0]=(1-KG)*Vfdm[0][i];
        VD[y][i]=Vfd[0][i];
    }
    Varn=VN;
}

```

```

} /* down_scan */

```

```

left_scan(y,inc1,Varn,ws,Vfd,VL,A)
int y,inc1;
float Varn,ws,Vfd[4][4],A[4],VL[NC][4];
{
    int x,i,j,k,afunc[D];
    float fhat,fhm,ennov,m,KG,
        m11,m12,m13,m21,m22,m23,m31,m32,m33,
        d1,d2,d3,d4,lvar,
        Vfdmm[D][D],Vfdm[D][D];

    for(x=(NC-1-inc1);x>=inc1;x--){
        afunc[0]=pixel(x+1,y);afunc[1]=pixel(x+1,y+1);
        afunc[2]=pixel(x,y+1);afunc[3]=pixel(x-1,y+1);
        fhm=0.0;
        for(j=0;j<D;j++) fhm += A[j]*afunc[j];
        enn timer=(float) pixel(x,y)-fhm;
    }
}

```



```

if((y<NR-4 && x>=4 && x<NC-4)){
    m=mean(x,y);
    lvar=var(x,y,m);
    lvar -=VN;
    if(lvar>VN) Varn = VN/2.0;
    else Varn=VN;
    if(lvar>VN+50.0){
        m11=local_mean(x-3,y-3);
        m12=local_mean(x,y-3);
        m13=local_mean(x+3,y-3);
        m21=local_mean(x-3,y);
        m22=local_mean(x,y);
        m23=local_mean(x+3,y);
        m31=local_mean(x-3,y+3);
        m32=local_mean(x,y+3);
        m33=local_mean(x+3,y+3);
        d1=fabs(m11-m33)+fabs(m12-m23)+fabs(m21-m32);
        d2=fabs(m11-m31)+fabs(m12-m32)+fabs(m13-m33);
        d3=fabs(m12-m21)+fabs(m13-m31)+fabs(m23-m32);
        d4=fabs(m13-m11)+fabs(m23-m21)+fabs(m33-m31);

        if(d1>d2 && d1>d3 && d1>d4){
            if(fabs(m11-m22)<=fabs(m22-m33)) lvar=local_var(x-3,y-3,m11);
            else lvar=local_var(x+3,y+3,m33);
        }
        if(d2>d1 && d2>d3 && d2>d4){
            if(fabs(m12-m22)<=fabs(m22-m32)) lvar=local_var(x,y-3,m12);
            else lvar=local_var(x,y+3,m32);
        }
        if(d3>d1 && d3>d2 && d3>d4){
            if(fabs(m13-m22)<=fabs(m22-m31)) lvar=local_var(x+3,y-3,m13);
            else lvar=local_var(x-3,y+3,m31);
        }
        if(d4>d1 && d4>d2 && d4>d3){
            if(fabs(m23-m22)<=fabs(m22-m21)) lvar=local_var(x,y+3,m23);
            else lvar=local_var(x,y-3,m21);
        }
        lvar -=VN;
        if(lvar>0.0) Varn = VN*exp(-200.0*lvar);
        else Varn=VN;
    }
}

if((y<NR-1) && (x>1) && (x<NC-1))
    for(i=1;i<D;i++){
        if(i==1)
            for(j=i;j<D;j++) Vfd[i][j]=Vfd[j][i]=VL[x+1][j-i];
        if(i==2)
            for(j=i;j<D;j++) Vfd[i][j]=Vfd[j][i]=VL[x][j-i];
        if(i==3)

```

```

        for(j=i;j<D;j++) Vfd[i][j]=Vfd[j][i]=VL[x-1][j-i];
    }

    for(i=0;i<D;i++){
        for(j=0;j<D;j++){
            if(i==0){
                Vfdmm[0][j]=0.0;
                for(k=0;k<D;k++) Vfdmm[0][j] += A[k]*Vfd[k][j];
            }
            else Vfdmm[i][j]=Vfd[i][j];
        }
    }
    for(i=0;i<D;i++){
        for(j=0;j<D;j++){
            if(j==0){
                Vfdm[i][j]=0.0;
                for(k=0;k<D;k++) Vfdm[i][j] += Vfdmm[i][k]*A[k];
            }
            else Vfdm[i][j]=Vfd[i][j];
        }
    }
    Vfdm[0][0] += ws;
    KG = Vfdm[0][0]/(Vfdm[0][0]+Varn);
    fhat=fhm + KG*ennov;
    if(fhat<=0.0) fhat=0.0;
    if(fhat>=255.0) fhat=255.0;
    plot(x,y,(int) fhat);
    for(i=0;i<D;i++){
        Vfd[0][i]=Vfd[i][0]=(1-KG)*Vfdm[0][i];
        VL[x][i]=Vfd[0][i];
    }
    Varn=VN;
}

} /* left_scan */

```

```

up_scan(x,inc2,Varn,ws,Vfd,VU,A)
int x,inc2;
float Varn,ws,Vfd[4][4],A[4],VU[NR][4];
{
    int y,i,j,k,afunc[D];
    float fhat,fhm,ennov,m,KG,
        m11,m12,m13,m21,m22,m23,m31,m32,m33,
        d1,d2,d3,d4,lvar,
        Vfdmm[D][D],Vfdm[D][D];

    for(y=(NR-2-inc2).y>inc2;y--){
        afunc[0]=pixel(x,y+1);afunc[1]=pixel(x-1,y+1);
        afunc[2]=pixel(x-1,y);afunc[3]=pixel(x-1,y-1);
        fhm=0.0;
        for(j=0;j<D;j++) fhm += A[j]*afunc[j];
        ennov=(float) pixel(x,y)-fhm;
    }
}

```

```

if(y>=4 && x>=4 && y<NR-4){
    m=mean(x,y);
    lvar=var(x,y,m);
    lvar -=VN;
    if(lvar>VN) Varn = VN/2.0;
    else Varn=VN;
    if(lvar>VN+50.0){
        m11=local_mean(x-3,y-3);
        m12=local_mean(x,y-3);
        m13=local_mean(x+3,y-3);
        m21=local_mean(x-3,y);
        m22=local_mean(x,y);
        m23=local_mean(x+3,y);
        m31=local_mean(x-3,y+3);
        m32=local_mean(x,y+3);
        m33=local_mean(x+3,y+3);
        d1=fabs(m11-m33)+fabs(m12-m23)+fabs(m21-m32);
        d2=fabs(m11-m31)+fabs(m12-m32)+fabs(m13-m33);
        d3=fabs(m12-m21)+fabs(m13-m31)+fabs(m23-m32);
        d4=fabs(m13-m11)+fabs(m23-m21)+fabs(m33-m31);

        if(d1>d2 && d1>d3 && d1>d4){
            if(fabs(m11-m22)<=fabs(m22-m33)) lvar=local_var(x-3,y-3,m11);
            else lvar=local_var(x+3,y+3,m33);
        }
        if(d2>d1 && d2>d3 && d2>d4){
            if(fabs(m12-m22)<=fabs(m22-m32)) lvar=local_var(x,y-3,m12);
            else lvar=local_var(x,y+3,m32);
        }
        if(d3>d1 && d3>d2 && d3>d4){
            if(fabs(m13-m22)<=fabs(m22-m31)) lvar=local_var(x+3,y-3,m13);
            else lvar=local_var(x-3,y+3,m31);
        }
        if(d4>d1 && d4>d2 && d4>d3){
            if(fabs(m23-m22)<=fabs(m22-m21)) lvar=local_var(x,y+3,m23);
            else lvar=local_var(x,y-3,m21);
        }
        lvar -=VN;
        if(lvar>0.0 ) Varn = VN*exp(-200.0*lvar);
        else Varn=VN;
    }
}

if((y<NR-1) && (x>1) && (y>1))
    for(i=1;i<D;i++){
        if(i==1)
            for(j=i;j<D;j++) Vfd[i][j]=VU[y+1][j-i];
        if(i==2)
            for(j=i;j<D;j++) Vfd[i][j]=VU[y][j-i];
        if(i==3)

```

```

    for(j=i;j<D;j++) Vfd[i][j]=VU[y-1][j-i];
}

for(i=0;i<D;i++)
  for(j=0;j<D;j++){
    if(i==0){
      Vfdmm[0][j]=0.0;
      for(k=0;k<D;k++) Vfdmm[0][j] += A[k]*Vfd[k][j];
    }
    else Vfdmm[i][j]=Vfd[i][j];
  }
for(i=0;i<D;i++)
  for(j=0;j<D;j++){
    if(j==0){
      Vfdm[i][j]=0.0;
      for(k=0;k<D;k++) Vfdm[i][j] += Vfdmm[i][k]*A[k];
    }
    else Vfdm[i][j]=Vfd[i][j];
  }
Vfdm[0][0] += ws;
KG = Vfdm[0][0]/(Vfdm[0][0]+Varn);
fhat=fhm + KG*ennov;
if(fhat<=0.0) fhat=0.0;
if(fhat>=255.0) fhat=255.0;
plot(x,y,(int) fhat);
for(i=0;i<D;i++){
  Vfd[0][i]=Vfd[i][0]=(1-KG)*Vfdm[0][i];
  VU[y][i]=Vfd[0][i];
}
Varn=VN;
}

} /* up_scan */

```

```
/*.....
```

ROBALG.C

This is the robust algorithm proposed by Kashyap and Eom for parameter estimation and restoration of images corrupted by a combination of both Gaussian noise and impulse noise (outliers).

```
...../
```

```
#include <stdio.h>      /* standard I/O header file */
#include <math.h>        /* math header file */
#include <pipdrv.h>      /* utilities header file */
```

```
#define NC    512
#define NR    420
#define D     3
#define mside1 8
```

```
static float sqrg;
#define SQR(a) (sqrg=(a), sqrg*sqrg)
unsigned char w[mside1][mside1],afunc[D],
              w1[mside1][mside1],buffer[mside1];
```

```
FILE *sfp,*fp;
```

```
main()
{
```

```
int x,y,i,j,k,li,ki,K,
    xstart,ystart,msid1,msid2;
long pos;
```

```
float sum,resid,msize,NPT,Varn,wm,wt,temp,
      sign,chisq,chisqrt,a,b,c,ics,fhat,fhm,ennov,
      beta[D],C[D][2*D],covar[D][D],
      icovar[D][D],resid1[D],T[D],A[D];
```

```
ystart=0;
a=1.5;
b=2.5;
c=3.5;
```

```
if((fp=fopen("c:\\hosni\\girl5.pic","rb"))!=NULL)
do{
for(xstart=0;xstart<NC-2;xstart += (mside1-1)){
if((NC-1-xstart)>mside1) msid1=mside1;
else msid1=(NC-1-xstart);
```

```

    if((NR-1-ystart)>=msid1) msid2=msid1;
    else msid2=(NR-1-ystart);
    NPT=(float) (msid2-1)*(msid1-1);
    msize=(float) (msid1*msid2);
    for(y=ystart;y<(ystart+msid2);y++){
        printf("y= %d\r",y);
        pos= (y*512L+xstart);
        fseek(fp,pos,0);
        fread(buffer,1,msid1,fp);
        for(x=0;x<msid1;x++) w[(y-ystart)][x]=w1[(y-ystart)][x]=buffer[x];
    }/* for y */

    if(xstart>0){
        for(k=0;k<msid1;k++) w[k][0]=pixel(xstart,(k+ystart));
    }
    if(ystart>0)
        for(k=0;k<msid1;k++) w[0][k]=pixel((xstart+k),ystart);

    for(i=0;i<D;i++){
        beta[i]=0.0;
        for(j=0;j<D;j++) covar[i][j]=0.0;
    }

    for(i=1;i<msid2;i++){
        for(j=1;j<msid1;j++){
            afunc[0]=w[i][j-1];
            afunc[1]=w[i-1][j-1];
            afunc[2]=w[i-1][j];
            wm=(float) w[i][j];
            for(l=0;l<D;l++){
                wt= (float) afunc[l];
                for(k=0;k<=l;k++) covar[l][k] += wt*afunc[k];
                beta[l] += wm*wt;
            }
        }
    }

    for(j=1;j<D;j++)
        for(k=0;k<=j-1;k++) covar[k][j]=covar[j][k];

    for(i=0;i<D;i++){
        for(j=0;j<D;j++){
            C[i][j]=covar[i][j];
            C[i][j+D]=0.0;
            if(i==j) C[i][j+D]=1.0;
        }
    }
    for(ii=0;ii<=1;ii++){
        for(k=0;k<D;k++){
            for(l=2*D-1;l>=k;l--){
                C[k][l]=C[k][l]/C[k][k];
            }
        }
    }

```

```

    if(k==D-1) goto ten;
    for(j=k+1;j<D;j++){
        for(i=2*D-1;i>=k+1;--i){
            if(ii==0) C[j][i]=C[j][i]-C[j][k]*C[k][i];
            if(ii==1) C[k][i]=C[k][i]-C[k][j]*C[j][i];
        }
    }
    ten :
        continue;
}
}
for(i=0;i<D;i++){
    A[i]=0.0;
    for(j=0;j<D;j++){
        icovar[i][j]=C[i][j+D];
        A[i] += icovar[i][j]*beta[j];
    }
}

chisq=0.0;
for(i=1;i<msid2;i++){
    for(j=1;j<msid1;j++){
        afunc[0]=w[i][j-1];
        afunc[1]=w[i-1][j-1];
        afunc[2]=w[i-1][j];
        for(l=0,sum=0.0;l<D;l++) sum += A[l]*afunc[l];
        chisq += SQR(w[i][j]-sum);
    }
}
chisq /=NPT;
chisqrt= sqrt(chisq);
printf("\n\t%.2f\n",chisq);

kk=0;
do{

for(y=1;y<msid2;y++){
    printf("y=%d\r",y);
    for(x=1;x<msid1;x++){
        afunc[0]=w[y][x-1];
        afunc[1]=w[y-1][x-1];
        afunc[2]=w[y-1][x];
        fhm=0.0;
        for(j=0;j<D;j++) fhm += afunc[j]*A[j];
        fhm +=1.0;
        ennov=(float) w[y][x]-fhm;
        resid = ennov/chisqrt;
        if(resid>0.0) sign=1.0;
        else sign=-1.0;

        if(fabs(resid)<=a) resid=resid;
    }
}
}

```

```

else if(fabs(resid)<=b && fabs(resid)>a) resid = a*sign;
else if(fabs(resid)>b && fabs(resid)<=c)
    resid=(c-fabs(resid))/(c-b)*a*sign;
else resid=0.0;

resid *= chisqrt;
fhat = fhm + resid;
if(fhat>255.0) fhat = 255.0;
if(fhat<0.0) fhat = 0.0;
plot((x+xstart),(y+ystart),(int) fhat);
w1[y][x]=(int) fhat;

} /* for x */
} /* for y */

K=1;
do{

temp=chisqrt;
res=0.0;
for(i=0;i<D;i++) resid1[i]=0.0;
for(i=1;i<msid2;i++){
    for(j=1;j<msid1;j++){
        afunc[0]=w1[i][j-1];
        afunc[1]=w1[i-1][j-1];
        afunc[2]=w1[i-1][j];
    }
    for(k=0,fhm=0.0;k<D;k++) fhm += A[k]*afunc[k];
    fhm += 1.0;
    ennov=w1[i][j]-fhm;
    resid = ennov/chisqrt;
    if(resid>0.0) sign=1.0;
    else sign=-1.0;
    if(fabs(resid)<=a) resid=resid;
    else if(fabs(resid)<=b && fabs(resid)>a) resid = a*sign;
    else if(fabs(resid)>b && fabs(resid)<=c)
        resid=(c-fabs(resid))/(c-b)*a*sign;
    else resid=0.0;
    resid *= chisqrt;
    res += SQR(resid);
    for(k=0;k<D;k++) resid1[k] += resid*afunc[k];
}
}

chisq =res/NPT;
chisqrt=sqrt(chisq);
printf("\n\t%.2f\n",chisq);

for(j=0;j<D;j++)
    for(k=0;k<D;k++) covar[j][k]=0.0;
for(i=1;i<msid2;i++)
    for(j=1;j<msid1;j++){
        afunc[0]=w1[i][j-1];

```



```

    afunc[1]=w1[i-1][j-1];
    afunc[2]=w1[i-1][j];
    for(l=0;l<D;l++){
        wt= (float) afunc[l];
        for(k=0;k<=l;k++) covar[l][k] += wt*afunc[k];
    }
}

for(j=1;j<D;j++){
    for(k=0;k<=j-1;k++) covar[k][j]=covar[j][k];

for(i=0;i<D;i++){
    for(j=0;j<D;j++){
        C[i][j]=covar[i][j];
        C[i][j+D]=0.0;
        if(i==j) C[i][j+D]=1.0;
    }
for(ii=0;ii<=1;ii++){
    for(k=0;k<D;k++){
        for(l=2*D-1;l>=k;l--){
            C[k][l]=C[k][l]/C[k][k];
        }
        if(k==D-1) goto tnt;
        for(j=k+1;j<D;j++){
            for(i=2*D-1;i>=k+1;i--){
                if(ii==0) C[j][i]=C[j][i]-C[j][k]*C[k][i];
                if(ii==1) C[k][i]=C[k][i]-C[k][j]*C[j][i];
            }
        }
        tnt :
        continue;
    }
}

for(i=0;i<D;i++){
    for(j=0;j<D;j++) icovar[i][j]=C[i][j+D];

for(i=0;i<D;i++){
    T[i]=0.0;
    for(j=0;j<D;j++) T[i] += icovar[i][j]*resid1[j];
}
for(i=0;i<D;i++) A[i] += T[i];

K++;
} /* do */
while(fabs(chisqrt-temp)>=1.0 && K<=4);

for(i=1;i<msid2;i++)
    for(j=1;j<msid1;j++) w[i][j]=w1[i][j];
kk++;

```

```

} /* do */
while(kk<=3);

} /* for xstart */
ystart += (msid2-1);

}/* DO */
while(ystart<NR);

fclose(fp);
} /* main */

```

```

/*****

```

MEDIAN.C

This is a median filter using a window of size 3x3. The filter runs in the raster scanning direction for the restoration of noisy pictures.

```

*****/

```

```

#include <stdio.h>
#include <pipdrv.h>
#include <math.h>

#define NC 512
#define NR 420
#define N 3
#define NN N*N

FILE *sfp;

main(){

float *data, *data1, xmed, block[N][N];
int i,j,k,x,y;
long pos;
unsigned char buff[N*512];

sfp=fopen("\\hosni\\girl5.pic","rb");

```

```

for(y=0;y<NR;y++){
    pos=y*512L;
    fseek(sfp,pos,0);
    fread(buff,1,N*512L,sfp);
    if(y>=1)
        for(k=0;k<NC;k++) buff[k]=pixel(k,y);

    for(x=0;x<NC-N/2;x++){
        for(i=0;i<N;i++){
            for(j=0;j<N;j++){
                block[i][j]= (float) buff[i*512+x+j];
            }
        }

        for(i=0;i<N;i++)
            for(j=0;j<N;j++) data[N*i+j]=block[i][j];

/* if(x>N/2) data[N+1]=pixel(x,y+N/2);*/
        for(i=1;i<=NN;i++) data1[i]=data[i-1];
        mdian(data1,NN,&xmed);
/*    for(i=0;i<N;i++)
        for(j=0;j<N;j++)*/ plot(x,y+1,(int) xmed);

    } /* for x */
} /* for y */
fclose(sfp);

/* main */

mdian(x,num,xmed)
float x[], *xmed;
int num;
{
    int n2,n2p;
    void sort();

    sort(num,x);
    n2p=(n2=num/2) +1;
    *xmed=(num % 2 ? x[n2p] : 0.5*(x[n2]+x[n2p]));
}

void sort(n,ra)
int r;
float ra[];
{
    int l,j,ir,i;
    float rra;
    l=(n >> 1)+1;
    ir=n;
    for(;;){
        if(l>1) rra=ra[--l];

```

```

else{
    rra=ra[ir];
    ra[ir]=ra[1];
    if(--ir==1){
        ra[1]=rra;
        return;
    }
}
i=1;
j=1 << 1;
while(j<=ir){
    if(j<ir && ra[j]<ra[j+1]) ++j;
    if(rra<ra[j]){
        ra[i]=ra[j];
        j += (i=j);
    }
    else j=ir+1;
}
ra[i]=rra;
}
}/* sort */

```

```
#include <math.h>
```

```

#define BIG 1.0e30
#define AFAC 1.5
#define AMP 1.5

```

```
void mdian1(x,n,xmed)
```

```

int n;
float x[], *xmed;
{
    int np,nm,j;
    float xx, xp, xm, sumx, sum, eps, stemp, dum, ap, am, aa, a;

```

```

    a=0.5*(x[1]+x[n]);
    eps=fabs(x[n]-x[1]);
    am=-(ap=BIG);
    for(;;){
        sum=sumx=0.0;
        np=nm=0;
        xm=-(xp=BIG);
        for(j=1;j<=n;j++){
            xx=x[j];
            if(xx!=a){
                if(xx>a){
                    ++np;

```

```

        if(xx < xp) xp=xx;
    }
    else if(xx < a){
        ++nm;
        if(xx > xm) xm=xx;
    }
    sum += dum=1.0/(eps+fabs(xx-a));
    sumx += xx*dum;
}
}
stemp=(sumx/sum)-a;
if(np-nm >=2){
    am=a;
    aa=stemp < 0.0 ? xp : xp+stemp*AMP;
    if(aa > ap) aa=0.5*(a+ap);
    eps=AFAC*fabs(aa-a);
    a=aa;
}
else if(nm-np >=2){
    ap=a;
    aa=stemp > 0.0 ? xm : xm+stemp*AMP;
    if(aa < am) aa=0.5*(a+am);
    eps=AFAC*fabs(aa-a);
    a=aa;
}
else {
    if(n % 2 == 0){
        *xmed=0.5*(np==nm ? xp+xm : np > nm ? a+xp : xm+a);
    }
    else{
        *xmed = np == nm ? a : np > nm ? xp : xm;
    }
    return;
}
}
/* mduan1 */

```

```
/******
```

ROBKAL.C

This is a robust Kalman Filter routine which is a modified Reduced Update Kalman Filter for the restoration of images corrupted by an impulse contaminated white Gaussian noise. This routine is an iterative procedure that uses robust estimation method to determine the modeling coefficients and the variance of the drive noise and the observation noise.

```
*****/
```

```
#include <stdio.h>          /* standard I/O header file */
#include <math.h>            /* math header file */
#include <pipdrv.h>

#define NC    512          /* number of columns in the image */
#define NR    420          /* number of rows in the image array */
#define D      4
#define msid1  8
```

```
static float sqrrarg;
#define SQR(a) (sqrrarg=(a), sqrrarg*sqrrarg)
unsigned char w[msid1][msid1], afunc[D],
              w1[msid1][msid1], buffer[msid1];
```

```
FILE *sfp,*fp;
```

```
main()
{
```

```
int x,y,i,j,k,l,ii,kk,K,
    xstart,ystart,msid1,msid2;
long pos;
```

```
float sum,resid,msize,NPT,Varn,chisqrt,
      wm,wt,wt1,temp,res1,mean,ennov1,
      chisq,a,b,c,a1,b1,c1,sign,sign1,
      res,fhat,fhm,ennov,KG,residue,temp1,
      Vfdm[D][D],beta[D],C[D][2*D],covar[D][D],
      Vfdmm[D][D],Vfd0[D][D],Vfd[D][D],icovar[D][D],
      Vf[msid1][D],resid1[D],T[D],A[D];
```

```
ystart=0;
a=1.5; a1=1.0;
b=2.5; b1=1.5;
c=3.5; c1=2.5;
```

```

if((fp=fopen("c:\\hosni\\girl5.pic","rb"))!=NULL)
sfp=fopen("c:\\hosni\\girl.pic","rb");

do{
  for(xstart=0;xstart<NC-2;xstart += (mside1-2)){
    if((NC-1-xstart)>mside1) mside1=mside1;
    else mside1=(NC-1-xstart);
    if((NR-1-ystart)>=mside1) mside2=mside1;
    else mside2=(NR-1-ystart);
    NPT=(float) (msid2-1)*(msid1-1);
    msize=(float) (msid1*msid2);
    for(y=ystart;y<(ystart+msid2);y++){
      printf("y= %d\r",y);
      pos = (y*512L+xstart);
      fseek(fp,pos,0);
      fseek(sfp,pos,0);
      fread(buffer,1,msid1,fp);
      for(x=0;x<msid1;x++) w[(y-ystart)][x]= buffer[x];
      fread(buffer,1,msid1,sfp);
      for(x=0;x<msid1;x++) w1[(y-ystart)][x]= buffer[x];
    }/* for y */

    mean=0.0;
    for(i=0;i<msid2;i++){
      for(j=0;j<msid1;j++) mean += w1[i][j]/msize;

    if(xstart>0){
      for(k=0;k<msid1;k++) w[k][0]=pixel(xstart,(k+ystart));
    }
    if(ystart>0)
      for(k=0;k<msid1;k++) w[0][k]=pixel((xstart+k),ystart);

    for(i=0;i<D;i++){
      beta[i]=0.0;
      for(j=0;j<D;j++){
        covar[i][j]=0.0;
        Vfd[i][j]=0.0;
      }
    }

    for(i=1;i<msid2;i++){
      for(j=1;j<msid1-1;j++){
        afunc[0]=w[i][j-1];
        afunc[1]=w[i-1][j-1];
        afunc[2]=w[i-1][j];
        afunc[3]=w[i-1][j+1];
        wm=(float) w[i][j];
        for(l=0;l<D;l++){
          wt= (float) afunc[l];
          for(k=0;k<=l;k++) covar[l][k] += wt*afunc[k];
          beta[l] += wm*wt;
        }
      }
    }
  }
}

```

```

    }
  }
}

for(j=1;j<D;j++){
  for(k=0;k<=j-1;k++){ covar[k][j]=covar[j][k];

  for(i=0;i<D;i++){
    for(j=0;j<D;j++){
      C[i][j]=covar[i][j];
      C[i][j+D]=0.0;
      if(i==j) C[i][j+D]=1.0;
    }
    for(ii=0;ii<=1;ii++){
      for(k=0;k<D;k++){
        for(l=2*D-1;l>=k;l--){
          C[k][l]=C[k][l]/C[k][k];
        }
        if(k==D-1) goto ten;
        for(j=k+1;j<D;j++){
          for(i=2*D-1;i>=k+1;i--){
            if(ii==0) C[j][i]=C[j][i]-C[j][k]*C[k][i];
            if(ii==1) C[k][i]=C[k][i]-C[k][j]*C[j][i];
          }
        }
        ten :
          continue;
      }
    }
    for(i=0;i<D;i++){
      A[i]=0.0;
      for(j=0;j<D;j++){
        icovar[i][j]=C[i][j+D];
        A[i] += icovar[i][j]*beta[j];
      }
    }

    chisq=0.0;
    for(i=1;i<msid2;i++){
      for(j=1;j<msid1-1;j++){
        afunc[0]=w[i][j-1];
        afunc[1]=w[i-1][j-1];
        afunc[2]=w[i-1][j];
        afunc[3]=w[i-1][j+1];
        for(l=0,sum=0.0;l<D;l++) sum += A[l]*afunc[l];
        chisq += SQR(w[i][j]-sum);
      }
    }
    chisq /=NPT;
    chisqrt=sqrt(chisq);
    for(y=1;y<msid2;y++){

```



```

for(x=1;x<msid1-1;x++){
    afunc[0]=w[y][x-1];
    afunc[1]=w[y-1][x-1];
    afunc[2]=w[y-1][x];
    afunc[3]=w[y-1][x+1];
    for(j=0,fhm=0.0;j<D;j++) fhm += afunc[j]*A[j];
    fhm +=1.0;
    ennov=(float) w[y][x]-fhm;
    resid = (float)(ennov)/chisqrt;
    if(ennov>0) sign=1.0;
    else sign=-1.0;
    if(fabs(resid)<=a) resid=resid;
    else if(fabs(resid)<=b && fabs(resid)>a) resid = a*sign;
    else if(fabs(resid)>b && fabs(resid)<=c)
        resid=(c-fabs(resid))/(c-b)*a*sign;
    else resid=0.0;
    resid = resid*chisqrt;
    fhat=fhm + resid;
    if(fhat>255.0) fhat = 255.0;
    if(fhat<0.0) fhat = 0.0;
    plot((x-xstart),(y-ystart),(int) fhat);
    w1[y][x]=(int) fhat;
}
}

for(i=0;i<msid2;i++)
    for(j=0;j<msid1;j++) w[i][j]=w1[i][j];

K=1;
do{
    temp=chisqrt;
    res=0.0;
    for(i=0;i<D;i++) resid1[i]=0.0;
    for(y=1;y<msid2;y++){
        for(x=1;x<msid1-1;x++){
            afunc[0]=w[y][x-1];
            afunc[1]=w[y-1][x-1];
            afunc[2]=w[y-1][x];
            afunc[3]=w[y-1][x+1];
            for(j=0,fhm=0.0;j<D;j++) fhm += afunc[j]*A[j];
            fhm +=1.0;
            ennov=(float) w[y][x]-fhm;
            resid = (float)(ennov)/chisqrt;
            if(ennov>0) sign=1.0;
            else sign=-1.0;
            if(fabs(resid)<=a) resid=resid;
            else if(fabs(resid)<=b && fabs(resid)>a) resid = a*sign;
            else if(fabs(resid)>b && fabs(resid)<=c)
                resid=(c-fabs(resid))/(c-b)*a*sign;
            else resid=0.0;
            resid = resid*chisqrt;

```

```

res += SQR(resid);
for(k=0;k<D;k++) resid1[k] += resid*afunc[k];
}
:
chisq=res/NPT;
chisqrt=sqrt(chisq);

for(j=0;j<D;j++)
  for(k=0;k<D;k++) covar[j][k]=0.0;
for(i=1;i<msid2;i++)
  for(j=1;j<msid1-1;j++){
    afunc[0]=w[i][j-1];
    afunc[1]=w[i-1][j-1];
    afunc[2]=w[i-1][j];
    afunc[3]=w[i-1][j+1];
    for(l=0;l<D;l++){
      w1=(float) afunc[l];
      for(k=0;k<=l;k++) covar[l][k] += w1*afunc[k];
    }
  }

for(j=1;j<D;j++)
  for(k=0;k<=j-1;k++) covar[k][j]=covar[j][k];
for(i=0;i<D;i++){
  for(j=0;j<D;j++){
    C[i][j]=covar[i][j];
    C[i][j+D]=0.0;
    if(i==j) C[i][j+D]=1.0;
  }
  for(ii=0;ii<=1;ii++){
    for(k=0;k<D;k++){
      for(l=2*D-1;l>=k;l--){
        C[k][l]=C[k][l]/C[k][k];
      }
      if(k==D-1) goto tn;
      for(j=k+1;j<D;j++){
        for(i=2*D-1;i>=k+1;--i){
          if(ii==0) C[j][i]=C[j][i]-C[j][k]*C[k][i];
          if(ii==1) C[k][i]=C[k][i]-C[k][j]*C[j][i];
        }
      }
      tn :
      continue;
    }
  }
}

for(i=0;i<D;i++)
  for(j=0;j<D;j++) icovar[i][j]=C[i][j+D];

for(i=0;i<D;i++){
  T[i]=0.0;

```

```

    for(j=0;j<D;j++) T[i] += icovar[i][j]*resid1[j];
  }
  for(i=0;i<D;i++) A[i] += T[i];
  K++;
  printf("\t%.2f\n",chisq);
} /* do */
while(fabs(chisqr-temp)>1.0 && K<=4);

for(i=1;i<msid2;i++)
  for(j=1;j<msid1-1;j++){
    afunc[0]=w[i][j-1];
    afunc[1]=w[i-1][j-1];
    afunc[2]=w[i-1][j];
    afunc[3]=w[i-1][j+1];
    for(l=0;l<D;l++){
      wt= (float) (afunc[l]-mean);
      for(k=0;k<=l;k++) Vfd[l][k] += wt*(afunc[k]-mean);
    }
  }

for(j=1;j<D;j++)
  for(k=0;k<=j-1;k++) Vfd[k][j]=Vfd[j][k];

for(j=0;j<D;j++)
  for(k=0;k<D;k++) Vfd[j][k]=Vfd[j][k]/NPT;

for(i=0;i<msid1;i++)
  for(j=0;j<D;j++) Vf[i][j]=Vfd[0][j];

Varn=100.0;

kk=0;
do{

  for(i=0;i<D;i++) resid1[i]=0.0;
  res=0.0;
  res1=0.0;

  for(y=1;y<msid2;y++){
    printf("y=%zd\r",y);
    for(x=1;x<msid1-1;x++){
      afunc[0]=w[y][x-1];
      afunc[1]=w[y-1][x-1];
      afunc[2]=w[y-1][x];
      afunc[3]=w[y-1][x+1];
      for(j=0,fhm=0.0;j<D;j++) fhm += afunc[j]*A[j];
      fhm +=1.0;
      ennov=(float) w[y][x]-fhm;

      if((y>1) && (x<msid1-1) && (x>1)){
        for(j=0;j<D;j++){

```

```

    Vfd{0}[j] = Vfd{0}[j];
    Vfd{1}[j] = Vfl{x-1}[j];
    Vfd{2}[j] = Vfl{x}[j];
    Vfd{3}[j] = Vfl{x+1}[j];
  }
}

for(i=1;i<D;i++)
  for(j=0;j<=i;j++) Vfd[j][i]=Vfd[i][j];

for(i=0;i<D;i++)
  for(j=0;j<D;j++){
    if(i==0){
      Vfdmm[i][j]=0.0;
      for(k=0;k<D;k++) Vfdmm[i][j] += A[k]*Vfd[k][j];
    }
    else Vfdmm[i][j]=Vfd[i][j];
  }
for(i=0;i<D;i++)
  for(j=0;j<D;j++){
    if(j==0){
      Vfdm[i][j]=0.0;
      for(k=0;k<D;k++) Vfdm[i][j] += Vfdmm[i][k]*A[k];
    }
    else Vfdm[i][j]=Vfdmm[i][j];
  }

Vfdm[0][0] += chisq;
KG=Vfdm[0][0]/(Vfdm[0][0]+Varn);
for(i=0;i<D;i++){
  Vfd[0][i]=(1-KG)*Vfdm[0][i];
  Vfl[x][i]=Vfd[0][i];
}
fhat=fhm + KG*ennov+0.55;

if(fhat>255.0) fhat = 255.0;
if(fhat<0.0) fhat = 0.0;

ennov1=fhat-fhm;
residue = (ennov1)/sqrt(Varn);
if(ennov1>0) sign1=1.0;
else sign1=-1.0;

if(fabs(residue)<=a1) residue=residue;
else if(fabs(residue)<=b1 && fabs(residue)>a1) residue = a1*sign1;
else if(fabs(residue)>b1 && fabs(residue)<=c1)
  residue=(c1-fabs(residue))/(c1-b1)*a1*sign1;
else residue=0.0;

residue = residue*sqrt(Varn);
res1 += SQR(residue);

```

```

    plot((x+xstart),(y+ystart),(int) fhat);
    w[y][x]=(int) fhat;

    } /* for x */
} /* for y */

temp=chisq;
temp1=Varn;
Varn =res1/NPT;

printf("\t\t\t%.2f\n",Varn);

K=1;
do{
chisqrt=sqrt(chisq);
temp=chisqrt;
res=0.0;
for(i=0;i<D;i++){ resid1[i]=0.0;
for(y=1;y<msid2;y++){
    for(x=1;x<msid1-1;x++){
        afunc[0]=w[y][x-1];
        afunc[1]=w[y-1][x-1];
        afunc[2]=w[y-1][x];
        afunc[3]=w[y-1][x+1];
        for(j=0,fhm=0.0;j<D;j++) fhm += afunc[j]*A[j];
        fhm +=1.0;
        cnnov=(float) w[y][x]-fhm;
        resid = (float)(cnnov)/chisqrt;
        if(cnnov>0) sign=1.0;
        else sign=-1.0;
        if(fabs(resid)<=a) resid=resid;
        else if(fabs(resid)<=b && fabs(resid)>a) resid = a*sign;
        else if(fabs(resid)>b && fabs(resid)<=c)
            resid=(c-fabs(resid))/(c-b)*a*sign;
        else resid=0.0;
        resid = resid*chisqrt;
        res += SQR(resid);
    }
}
chisq=res/NPT;
chisqrt=sqrt(chisq);

for(j=0;j<D;j++){
    for(k=0;k<D;k++) covar[j][k]=0.0;
for(i=1;i<msid2;i++){
    for(j=1;j<msid1-1;j++){
        afunc[0]=w[i][j-1];
        afunc[1]=w[i-1][j-1];
        afunc[2]=w[i-1][j];

```

```

    afunc[3]=w[i-1][j+1];
    for(l=0;l<D;l++){
        wt= (float) afunc[l];
        for(k=0;k<=l;k++) covar[l][k] += wt*afunc[k];
    }
}

for(j=1;j<D;j++){
    for(k=0;k<=j-1;k++) covar[k][j]=covar[j][k];
    for(i=0;i<D;i++){
        for(j=0;j<D;j++){
            C[i][j]=covar[i][j];
            C[i][j+D]=0.0;
            if(i==j) C[i][j+D]=1.0;
        }
        for(ii=0;ii<=1;ii++){
            for(k=0;k<D;k++){
                for(l=2*D-1;l>=k;l--){
                    C[k][l]=C[k][l]/C[k][k];
                }
                if(k==D-1) goto tnn;
                for(j=k+1;j<D;j++){
                    for(i=2*D-1;i>=k+1;--i){
                        if(ii==0) C[j][i]=C[j][i]-C[j][k]*C[k][i];
                        if(ii==1) C[k][i]=C[k][i]-C[k][j]*C[j][i];
                    }
                }
                tnn :
                continue;
            }
        }
    }

    for(i=0;i<D;i++){
        for(j=0;j<D;j++) icovar[i][j]=C[i][j+D];
    }

    for(i=0;i<D;i++){
        T[i]=0.0;
        for(j=0;j<D;j++) T[i] += icovar[i][j]*resid1[j];
    }
    for(i=0;i<D;i++) A[i] += T[i];
    K++;
    printf("\t%.2f\n",chisq);
} /* do */
while(fabs(chisqrt-temp)>1.0 && K<=4);

kk++;
} /* do */

while(kk<3);

} /* for xstart */

```

```
ystart += (msid2-2);
```

```
}/* DO */
```

```
while(ystart<NR-2);
```

```
fclose(fp);
```

```
fclose(sfp);
```

```
} /** main **/
```

END

03·03·92

FIN