USING KALMAN FILTERING METHODS FOR IMAGE RESTORATION

A Thesis Submitted to the
Graduate School of Engineering and Sciences of Dokuz Eylül University
In Partial Fulfillment of the Requirements for the Degree of Master of Science
in Electrical and Electronics Engineering

by

Mehmet Ali ARABACI

September, 2008
İZMİR

M.Sc THESIS EXAMINATION RESULT FORM

We have read the thesis entitled **"USING KALMAN FILTERING METHODS FOR IMAGE RESTORATION"** completed by **Mehmet Ali ARABACI** under supervision of **ASST. PROF. DR. OLCAY AKAY** and we certify that in our opinion it is fully adequate, in scope and in quality, as a thesis for the degree of Master of Science.

.....

Asst. Prof. Dr. Olcay AKAY

Supervisor

.....

.....

Prof. Dr. Cahit HELVACI
Director
Graduate School of Natural and Applied Sciences

ACKNOWLEDGMENTS

At the first place, I would like to thank my advisor Asst. Prof. Dr. Olcay AKAY for his supervision, advice, and guidance from the very early stage of this research work. Above all and when most needed, he provided me encouragement and support in various ways. I also would like to thank Ömer ORAL for his advice and crucial contributions, which made him the backbone of this research work and this thesis. I hope to keep up our collaboration in the future. Lastly, I thank my family for their never ending support and motivation.

Mehmet Ali ARABACI

USING KALMAN FILTERING METHODS FOR IMAGE RESTORATION

ABSTRACT

A common problem in image processing is the restoration of an image from a given corrupted version. This problem is generally known as image restoration. There are various approaches to solve this problem like using restoration models or linear filtering. In this thesis, one of the linear filtering methods named Kalman filtering has been used for image restoration. For this purpose, two different image restoration scenarios have been defined and two different versions of Kalman filtering methods have been used for each of the scenarios. In the first part of the thesis, a simple scalar Kalman filter method is used for an image that contains multiple frames. In the second part of the thesis, a two-dimensional (2-D) full-plane block Kalman filter is used to restore noisy images. Simulation results are compared with some of the other restoration techniques.

Keywords: Image restoration, linear dynamical systems, state-space model, Kalman filtering, scalar Kalman filtering, full-plane block Kalman filtering.

KALMAN SÜZGECİ KULLANARAK İMGE ONARIMI

ÖZ

Görüntü işlemede karşılaşılan yaygın problemlerden biri verilen bozuk bir görüntüyü kullanarak onarım yapmaktır. Bu problem genelde görüntü onarımı olarak adlandırılır. Görüntü onarımı problemini çözmek için onarım modellerinin ya da doğrusal süzgeçlerin kullanılması gibi bir çok yaklaşım vardır. Bu tezde, görüntü onarımı için doğrusal süzgeç yöntemlerinden biri olan Kalman süzgeci yöntemi kullanılmıştır. Bu amaçla, iki farklı görüntü onarım problemi tanımlanmış ve her bir problem için farklı Kalman süzgeci yöntemi kullanılmıştır. Tezin birinci bölümünde, çoklu çerçeveden oluşan bir görüntü için basit bir skalar Kalman süzgeci yöntemi uygulanmıştır. Tezin ikinci bölümünde ise gürültülü görüntüler üzerinde iki boyutlu tam düzlemli blok Kalman süzgeci yöntemi kullanılmıştır. Benzetim sonuçları görüntü onarımında kullanılan diğer yöntemlerden bir kaçıyla karşılaştırılmıştır.

Anahtar sözcükler: Görüntü onarımı, doğrusal dinamik sistemler, durum-uzay modeli, Kalman süzgeci, skalar Kalman süzgeci, tam düzlemli blok Kalman süzgeci.

CONTENTS

Page

CHAPTER ONE

INTRODUCTION

1.1 Image Restoration

Image restoration aims to recover an image that has been corrupted or degraded. Therefore, all the improved techniques used in image restoration are used to eliminate or minimize the corruptions or degradations on images. Most of the time, image restoration is confused with the image enhancement. Although, there are common areas, image enhancement is largely a subjective process as compared to the image restoration. In fact, image enhancement techniques are mostly used to get a better visualization, extracting of image features or to manipulate an image in order to apply a predefined process. On the other hand, image restoration is used for eliminating degradations. Image restoration problems can be quantified precisely, whereas enhancement criteria are difficult to represent mathematically. Consequently, restoration techniques often depend only on the class or ensemble properties of a data set, whereas image enhancement techniques are much more image dependent.

Degradations may be caused by physical phenomena or the problems of sensing environment such as random atmospheric turbulence, sensor noise, camera misfocus, relative object-camera motion. Therefore, optics, electro-optics and electronics have a connection with the image restoration, because of their relation with the image acquiring processes. Basically, the first problem in image restoration is to model the optic and physical environment considering the image acquiring process. After finding an efficient representative model of the process, the inverse filtering method is applied in order to recover the original image. Consequently, the effectiveness of image restoration filters depends on the extent and accuracy of the knowledge of the degradation process as well as on the filter design criterion (Jain, 1988).

There are a lot of techniques and models used in image restoration. Any of these techniques can be applied to the images for image restoration by considering their

effectiveness, restrictions and complexities. Figure 1.1 shows widely used models in image restoration and the techniques according to whether it is a linear filtering method or not.

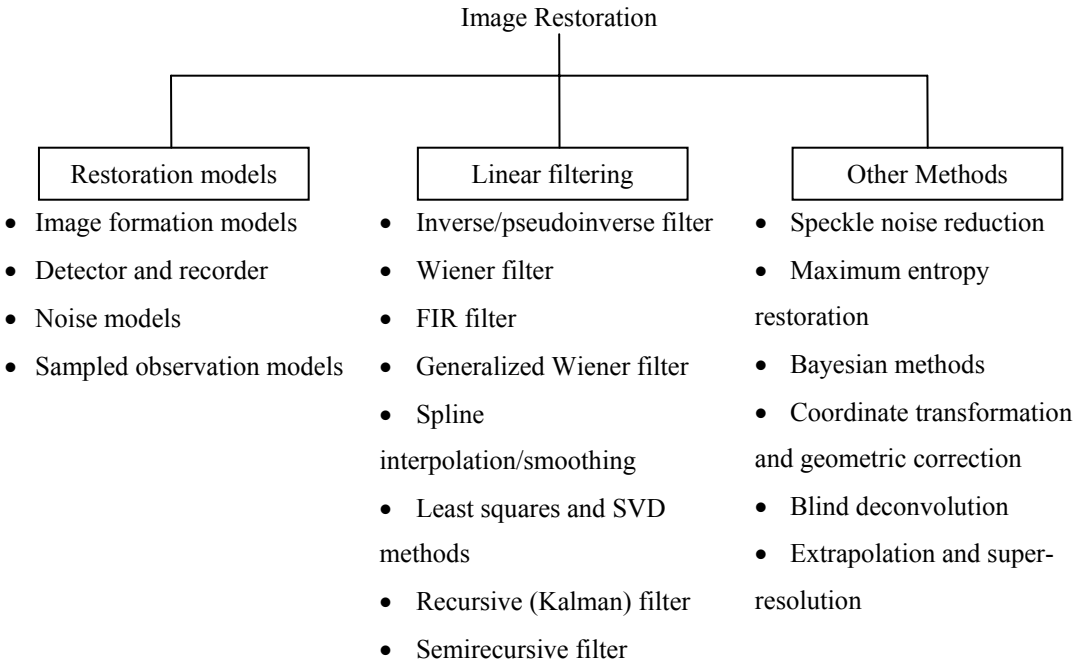


Figure 1.1 Hierarchy of image restoration.

As can be seen from Figure 1.1, Kalman filter is one of the available techniques used in image restoration. Although, it is considered as a linear filtering method, it can also be applied to nonlinear image models by using Extended Kalman filter formulation.

1.2 Introduction to Kalman Filter

Kalman filter is simply an *optimal recursive data processing algorithm*. There are many ways of defining *optimal*, dependent upon the criteria chosen to evaluate performance. One aspect of this optimality is that Kalman filter incorporates all information that can be provided to it. It processes all available measurements, regardless of their precision, to estimate the current value of the variables of interest, with use of knowledge of the process and measurement device dynamics, the

statistical description of the process noise, measurement errors, and uncertainty of dynamic models, and any available information about initial conditions of the variables of interest.

The word *recursive* in the previous description means that, unlike certain data processing concepts, Kalman filter does not require all previous data to be kept in storage and reprocessed every time a new measurement is taken. This will be of vital importance to the practically of filter implementation. The filter is actually a *data processing algorithm*. Despite the typical connotation of a filter as a black box containing electrical networks, the fact is that in most practical applications, the filter is just a computer program in a central processor. As such, it inherently incorporates discrete-time measurement samples rather than continuous-time inputs (Maybeck, 1979).

The Figure 1.2 depicts a typical situation in which Kalman filter could be used advantageously. A system of some sort is driven by some known controls, and measuring devices provide the value of certain pertinent quantities. Knowledge of these system inputs and outputs is all that is explicitly available from the physical system for estimation purposes.

The parameters called as *controls* and *observed measurements* shown in Figure 1.2 are known variables. The other parameters, *system error sources* and *measurement error sources*, can be measured during the process and updated at each step. One of the important points of Kalman filter application is to model the system box. At this point, modeling of linear systems will be explained by using state-space model.

1.3 Discrete-Time Linear Systems

A *linear system* is a mathematical model of a system based on the use of a linear operator. Linear systems typically exhibit features and properties that are much simpler than the general, nonlinear case. As a mathematical abstraction or

idealization, linear systems find important applications in automatic control theory, signal processing, and telecommunications.

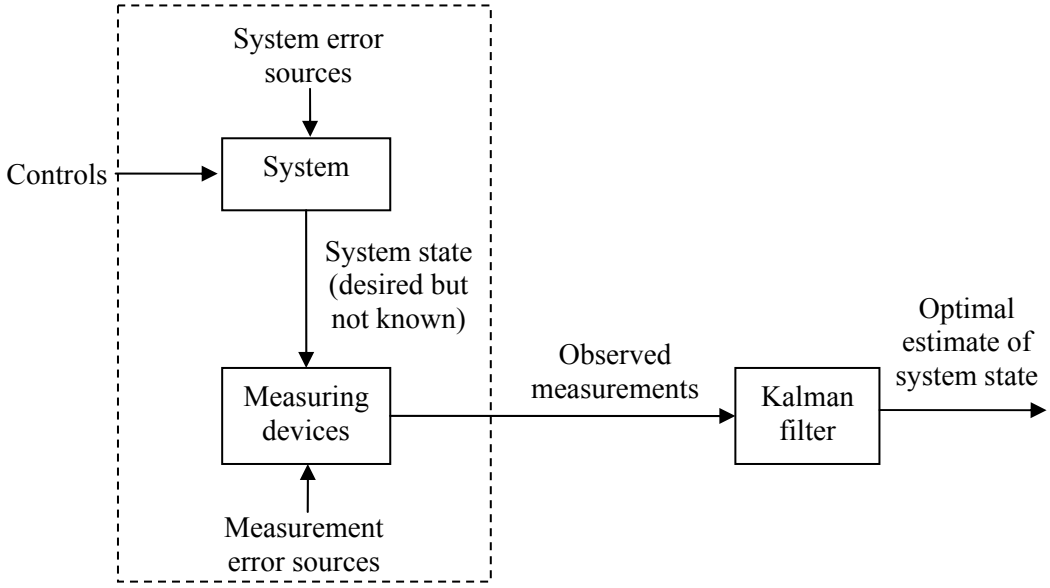


Figure 1.2 Typical Kalman filter application.

Typical differential equations of linear time-invariant systems are well adapted to analysis using the Laplace transform in the continuous case, and the Z-transform in the discrete case (especially in computer implementations).

The output of any discrete-time linear system is related to the input by the time-varying convolution sum;

$$y[n] = \sum_{k=-\infty}^{\infty} h[n,k]x[k], \tag{1.1}$$

or equivalently,

$$y[n] = \sum_{m=-\infty}^{\infty} h[n,n-m]x[n-m], \tag{1.2}$$

where $k = n - m$ represents the time lag between the stimulus at time m and the response at time n.

1.4 State Space Model of Discrete Time Linear Systems

In control engineering, a state space representation is a mathematical model of a physical system as a set of input, output and state variables related by first-order differential equations. To abstract from the number of inputs, outputs and states, the variables are expressed as vectors and the differential and algebraic equations are written in matrix form (the last one can be done when the dynamical system is linear and time-invariant). The state space representation (also known as the "time-domain approach") provides a convenient and compact way to model and analyze systems with multiple inputs and outputs.

The internal state variables are the smallest possible subset of system variables that can represent the entire state of the system at any given time. State variables must be linearly independent; a state variable cannot be a linear combination of other state variables. The minimum number of state variables required to represent a given system, *n*, is usually equal to the order of the system's defining differential equation. If the system is represented in transfer function form, the minimum number of state variables is equal to the order of the transfer function's denominator after it has been reduced to a proper fraction. It is important to understand that converting a state space realization to a transfer function form may lose some internal information about the system, and may provide a description of a system which is stable, when the state-space realization is unstable at certain points.

The most general discrete-time state space representation of a linear system with *p* inputs, *q* outputs and *n* state variables is written in the following form;

Discrete time-invariant:

$$x(k+1) = Ax(k) + Bu(k), \tag{1.3}$$

$$y(k) = Cx(k) + Du(k), \tag{1.4}$$

Discrete time-variant:

$$x(k+1) = A(k)x(k) + B(k)u(k), \tag{1.5}$$

$$y(k) = C(k)x(k) + D(k)u(k). \tag{1.6}$$

CHAPTER TWO

KALMAN FILTER THEORY

2.1 Estimation Problem

The problem we consider in this section is about estimating the state of a linear stochastic system by using measurements that are linear functions of the state.

We suppose that stochastic systems can be represented by plant and measurement discrete-time models as shown in Equations 2.1-2.4 in Table 2.1, with dimensions of the vector and matrix quantities shown in Table 2.2. The symbol $\Delta(k-l)$ stands for *Kronecker delta function* (Grewal & Andrews, 2001).

Table 2.1 Linear plant and measurement models.

Model	**Discrete Time**	**Equation Number**
Plant	$x_k = \Phi_{k-1} x_{k-1} + w_{k-1}$	(2.1)
Measurement	$z_k = H_k x_k + v_k$	(2.2)
Plant Noise	$E(w_k) = 0$ $E(w_k w_i^T) = \Delta(k-i) Q_k$	(2.3)
Observation Noise	$E(v_k) = 0$ $E(v_k v_i^T) = \Delta(k-i) R_k$	(2.4)

Table 2.2 Dimensions of vectors and matrices in linear model.

Symbol	**Dimension**	**Symbol**	**Dimension**
x, w	$n \times 1$	Φ, Q	$n \times n$
z, v	$\ell \times 1$	H	$\ell \times n$
R	$\ell \times \ell$	Δ	*scalar*

The measurement and plant noise v_k and w_k are assumed to be zero-mean Gaussian processes, and the initial value of the state x_0 is a Gaussian random variable with known mean, x_0 , and known error covariance matrix, P_0.

The objective will be to find an estimate of the n state vector x_k represented by a linear function of the measurements, $z_1, \ldots, z_k$, that minimize the weighted mean-square error;

$$E\left\langle [x_k - \hat{x}_k]^T M [x_k - \hat{x}_k] \right\rangle \tag{2.5}$$

where E represents the expected value and M is any *symmetric nonnegative-definite weighting matrix*.

The parameters $\Phi, H, Q,$ and R appearing in Equations 2.1-2.4 are called transition matrix, observation matrix, process noise covariance matrix, and measurement noise covariance matrix, respectively.

2.2 Kalman Filter

2.2.1 Estimator in Linear Form

Suppose that a measurement has been made at time t_k and that the information it provides is to be applied in updating the estimate of the state x of a stochastic system at time t_k. It is assumed that the measurement is linearly related to the state by an equation of the form $z_k = Hx_k + v_k$, where H is the *measurement sensitivity matrix* or the *observation matrix* and v_k is the measurement noise.

The optimal linear estimate is equivalent to the general (nonlinear) optimal estimator if the variables x and z are jointly Gaussian. Therefore, it suffices to seek an updated estimate $\hat{x}(+)$ based on the observation z_k that is a linear function of the a priori estimate and the measurement z ;

$$\hat{x}_k(+) = K_k^1 \hat{x}_k(-) + \overline{K}_k z_k , \tag{2.6}$$

where $\hat{x}_k(-)$ is the a priori estimate of x_k and $\hat{x}_k(+)$ is the a posteriori value of the estimate.

2.2.2 Optimization Problem

The matrices K_k^1 and $\bar{K}_k$ are as yet unknown. We seek those values of K_k^1 and $\bar{K}_k$ such that the new estimate $\hat{x}_k(+)$ will satisfy the orthogonality principle. This orthogonality condition can be written in the form

$$E\left\langle \left[x_k - \hat{x}_k(+) \right] z_i^T \right\rangle = 0, \quad i = 1,2,\ldots,k, \tag{2.7}$$

$$E\left\langle \left[x_k - \hat{x}_k(+) \right] z_k^T \right\rangle = 0. \tag{2.8}$$

where $E\langle\ \rangle$ shows the expectation.

If one substitutes the formula for x_k from Equation 2.1 (in Table 2.1) and for $\hat{x}_k(+)$ from Equation 2.6 into Equation 2.7, then one will observe from Equations 2.1 and 2.2 that data $z_1,\ldots,z_k$ do not involve the noise term w_k. Therefore, because the random sequences w_k and v_k are uncorrelated, it follows that $E\left(w_k z_i^T\right) = 0$ for $1 \le i \le k$.

Using this result, one can obtain the following relation,

$$E\left\langle \left[\Phi_{k-1} x_{k-1} + w_{k-1} - K_k^1 \hat{x}_k(-) - \bar{K}_k z_k \right] z_i^T \right\rangle = 0, \quad i = 1,2,\ldots,k-1. \tag{2.9}$$

Because $z_k = Hx_k + v_k$, Equation 2.9 can be rewritten as

$$E\left\langle \left[\Phi_{k-1} x_{k-1} - K_k^1 \hat{x}_k(-) - \bar{K}_k H_k x_k - \bar{K}_k v_k \right] z_i^T \right\rangle = 0. \tag{2.10}$$

We also know that Equations 2.7 and 2.8 hold at the previous step, that is,

$$E\left\langle \left[x_{k-1} - \hat{x}_{k-1}(+) \right] z_i^T \right\rangle = 0, \quad i = 1,2,\ldots,k-1$$

and

$$E\left\langle v_k z_i^T \right\rangle = 0, \quad i = 1,2,\ldots,k-1.$$

Equation 2.10 can be reduced to the form

$$\Phi_{k-1} E\left[x_{k-1} z_i^T \right] - K_k^1 E\left[\hat{x}_k(-) z_i^T \right] - \bar{K}_k H_k \Phi_{k-1} E\left[x_{k-1} z_i^T \right] - \bar{K}_k E\left[v_k z_i^T \right] = 0,$$

$$\Phi_{k-1} E\left[x_{k-1} z_i^T \right] - K_k^1 E\left[\hat{x}_k(-) z_i^T \right] - \bar{K}_k H_k \Phi_{k-1} E\left[x_{k-1} z_i^T \right] = 0,$$

$$E\left\langle \left[x_k - \bar{K}_k H_k x_k - K_k^1 x_k \right] - K_k^1 \left(\hat{x}_k(-) - x_k \right) \right\rangle z_i^T = 0,$$

$$\left[I - K_k^1 - \bar{K}_k H_k \right] E\left[x_k z_i^T \right] = 0. \tag{2.11}$$

Equation 2.11 can be satisfied for any given x_k if

$$K_k^1 = I - \bar{K}_k H_k. \tag{2.12}$$

Clearly, this choice of K_k^1 causes Equation 2.6 to satisfy a portion of the condition given by Equation 2.7. The choice of K_k^1 is such that Equation 2.8 is satisfied.

Let the errors be defined as

$$\tilde{x}_k(+) \triangleq \hat{x}_k(+) - x_k, \tag{2.13}$$

$$\tilde{x}_k(-) \triangleq \hat{x}_k(-) - x_k, \tag{2.14}$$

$$\tilde{z}_k \triangleq \tilde{z}_k(-) - z_k = H_k \hat{x}_k(-) - z_k. \tag{2.15}$$

where vectors $\tilde{x}_k(+)$ and $\tilde{x}_k(-)$ are the estimation errors after and before updates, respectively.

The parameter $\hat{x}_k$ depends linearly on x_k, which depends linearly on z_k. Therefore, from Equation 2.8, we have

$$E\left[x_k - \hat{x}_k(+) \right] z_k^T(-) = 0, \tag{2.16}$$

and also (by subtracting Equation 2.8 from Equation 2.16)

$$E\left[x_k - \hat{x}_k(+) \right] \tilde{z}_k^T = 0. \tag{2.17}$$

Substitute for x_k , $\hat{x}_k(+)$ and $\tilde{z}_k$ from Equations 2.1, 2.6, and 2.15, respectively. Then, we have

$$E\left[\Phi_{k-1}x_{k-1} + w_{k-1} - K_k^1(-) - \bar{K}_k z_k\right]\left[H_k\hat{x}_k(-) - z_k\right]^T = 0.$$

However, by system structure

$$E\left[w_k z_k^T\right] = E\left[w_k \hat{x}_k^T(+)\right] = 0,$$

$$E\left[\Phi_{k-1}x_{k-1} - K_k^1\hat{x}_k(-) - \bar{K}_k z_k\right]\left[H_k\hat{x}_k(-) - z_k\right]^T = 0.$$

Substituting for K_k^1, z_k ,and $\tilde{x}_k(-)$ and using the fact that $E\left[\tilde{x}_k(-)v_k^T\right] = 0$, this last result can be modified as follows;

$$0 = E\left\langle\left[\Phi_{k-1}x_{k-1} - \hat{x}_k(-) + \bar{K}_k H_k\hat{x}_k(-) - \bar{K}_k H_k x_k - \bar{K}_k v_k\right]\left[H_k\hat{x}_k(-) - H_k x_k - v_k\right]^T\right\rangle$$

$$= E\left\langle\left[\left(x_k - \hat{x}_k(-)\right) - \bar{K}_k H_k\left(x_k - \hat{x}_k(-)\right) - \bar{K}_k v_k\right]\left[H_k\tilde{x}_k(-) - v_k\right]^T\right\rangle$$

$$= E\left\langle\left[-\tilde{x}_k(-) + \bar{K}_k H_k\tilde{x}_k(-) - \bar{K}_k v_k\right]\left[H_k\tilde{x}_k(-) - v_k\right]^T\right\rangle.$$

By definition, the a priori covariance (the error covariance matrix before the update) is

$$P_k(-) = E\left[\tilde{x}_k(-)\tilde{x}_k^T(-)\right].$$

It satisfies the equation

$$\left[I - \bar{K}_k H_k\right]P_k(-)H_k^T - \bar{K}_k R_k = 0,$$

and therefore the gain can be expressed as

$$\bar{K}_k = P_k(-)H_k^T\left[H_k P_k(-)H_k^T + R_k\right]^{-1}, \tag{2.18}$$

which is the solution we seek for the gain as a function of the a priori covariance.

One can derive a similar formula for the a posteriori covariance (the error covariance matrix after update), which is defined as

$$P_k(+) = E\left[\tilde{x}_k(+)\tilde{x}_k^T(+)\right]. \tag{2.19}$$

By substituting Equation 2.12 into Equation 2.6, one obtains the equations

$$\hat{x}_k(+) = \left(I - \bar{K}_k H_k\right)\hat{x}_k(-) + \bar{K}_k z_k,$$

$$\hat{x}_k(+) = \hat{x}_k(-) + \bar{K}_k\left[z_k - H_k\hat{x}_k(-)\right]. \tag{2.20}$$

Subtract x_k from both sides of the latter equation and substitute z_k with Equation 2.2 to obtain the equations

$$\hat{x}_k(+) - x_k = \hat{x}_k(-) + \bar{K}_k H_k x_k + \bar{K}_k v_k - \bar{K}_k H_k \hat{x}_k(-) - x_k,$$

$$\tilde{x}_k(+) = \tilde{x}_k(-) - \bar{K}_k H_k \tilde{x}_k(-) + \bar{K}_k v_k. \tag{2.21}$$

By substituting Equation 2.21 into Equation 2.19 and noting that $E\left[\tilde{x}_k(-)v_k^T\right] = 0$, one obtains

$$P_k(+) = E\left\langle\left[I - \bar{K}_k H_k\right]\tilde{x}_k(-)\tilde{x}_k^T(-)\left[I - \bar{K}_k H_k\right]^T + \bar{K}_k v_k v_k^T \bar{K}_k^T\right\rangle$$

$$= \left(I - \bar{K}_k H_k\right)P_k(-)\left(I - \bar{K}_k H_k\right)^T + \bar{K}_k R_k \bar{K}_k^T. \tag{2.22}$$

By substituting for $\bar{K}_k$ from Equation 2.18, Equation 2.22 can be put in the following forms;

$$P_k(+) = P_k(-) - \bar{K}_k H_k P_k(-) - P_k(-)H_k^T\bar{K}_k^T + \bar{K}_k H_k P_k(-)H_k^T\bar{K}_k^T$$

$$+\bar{K}_k R_k \bar{K}_k^T$$

$$= \left(I - \bar{K}_k H_k\right)P_k(-) - P_k(-)H_k^T\bar{K}_k^T + \underbrace{\bar{K}_k\left(H_k P_k(-)H_k^T + R_k\right)}_{P_k(-)H_k^T}\bar{K}_k^T$$

$$= \left(I - \bar{K}_k H_k\right)P_k(-). \tag{2.23}$$

The last of which is the one most often used in computation. This implements the effect that *conditioning on the measurement* has on the covariance matrix of estimation uncertainty.

Error covariance extrapolation models the effects of time on the covariance matrix of estimation uncertainty, which is reflected in the a priori values of the covariance and state estimates

$$P_k(-) = E\left[\tilde{x}_k(-)\tilde{x}_k^T(-)\right],$$

$$\hat{x}_k(-) = \Phi_{k-1}\hat{x}_{k-1}(+), \tag{2.24}$$

respectively. Subtract x_k from both sides of the last equation to obtain the equations

$$\hat{x}_k(-) - x_k = \Phi_{k-1}\hat{x}_{k-1}(+) - x_k,$$

$$\tilde{x}_k(-) = \Phi_{k-1}\left[\hat{x}_{k-1}(+) - x_{k-1}\right] - w_{k-1},$$

$$= \Phi_{k-1}\tilde{x}_{k-1}(+) - w_{k-1}$$

for the propagation of the estimation error, $\tilde{x}$. Postmultiply it by $\tilde{x}_k^T(-)$ (on both sides of the equation) and take the expected values. Using the fact that $E\left[\tilde{x}_{k-1}w_{k-1}^T\right] = 0$, we obtain the results

$$P_k(-) = E\left[\tilde{x}_k(-)\tilde{x}_k^T(-)\right]$$

$$= \Phi_{k-1}E\left[\tilde{x}_{k-1}(+)\tilde{x}_{k-1}^T(+)\right]\Phi_{k-1}^T + E\left[w_{k-1}w_{k-1}^T\right]$$

$$= \Phi_{k-1}P_{k-1}(+)\Phi_{k-1}^T + Q_{k-1}, \tag{2.25}$$

which gives the a priori value of the covariance matrix of estimation uncertainty as a function of the previous a posteriori value.

2.3 Summary of Equations for the Discrete-Time Kalman Estimator

The equations derived in the previous section are summarized in Table 2.3. The relation of the filter to the system is illustrated in the block diagram of Figure 2.1. The basic steps of the computational procedure for the discrete-time Kalman estimator are as follows;

1. Compute $P_k(-)$ using $P_{k-1}(+)$, Φ_{k-1}, and Q_{k-1},
2. Compute $\bar{K}_k$ using $P_k(-)$ (computed in step 1), H_k, and R_k,
3. Compute $P_k(+)$ using $\bar{K}_k$ (computed in step 2) and $P_k(-)$ (from step 1),
4. Compute successive values of $\hat{x}_k(+)$ recursively using the computed values of $\bar{K}_k$ (from step 3), the given initial estimate $\hat{x}_0$, and the input data z_k.

Step 4 of the Kalman filter implementation (computation of $\hat{x}_k(+)$) can be implemented only for state vector propagation where simulator or real data sets are available.

In the design trade-offs, the covariance matrix update (steps 1 and 3) should be checked for symmetry and positive definiteness. Failure to attain either condition is a sign that something is wrong – either a program "bug" or an ill-conditioned problem. In order to overcome ill-conditioning, another equivalent expression for $P_k(+)$, called the "Joseph form" as shown in Equation 2.22 and given below,

$$P_k(+) = \left[I - \bar{K}_k H_k \right] P_k(-) \left[I - \bar{K}_k H_k \right]^T + \bar{K}_k R_k \bar{K}_k^T ,$$

can also be adopted.

Note that the right-hand side of the above equation is the summation of two symmetric matrices. The first of these is positive definite and the second is nonnegative definite, thereby making $P_k(+)$ a positive definite matrix.

There are many other forms for $\bar{K}_k$ and $P_k(+)$ that might not be as useful for robust computation. It can be shown that state vector update, Kalman gain, and error covariance equations represent an asymptotically stable system, and therefore, the estimate of state $\hat{x}_k$ becomes independent of the initial estimate $\hat{x}_0$, P_0 as k is increased (Grewal & Andrews, 2001).

Table 2.3 Discrete-time Kalman filter equations.

System dynamic model	$x_k = \Phi_{k-1} x_{k-1} + w_{k-1}$ $w_k \sim \mathrm{N}(0, Q_k)$
Measurement model	$z_k = H_k x_k + v_k$ $v_k \sim \mathrm{N}(0, R_k)$
Initial conditions	$E[x_0] = \hat{x}_0$ $E[\tilde{x}_0 \tilde{x}_0^T] = P_0$
Independence assumption	$E[w_k v_j^T] = 0$ for all k and j
State estimate extrapolation	$\hat{x}_k(-) = \Phi_{k-1} \hat{x}_{k-1}(+)$
Error covariance extrapolation	$P_k(-) = \Phi_{k-1} P_{k-1}(+) \Phi_{k-1}^T + Q_{k-1}$
State estimate observational update	$\hat{x}_k(+) = \hat{x}_k(-) + \bar{K}_k [z_k - H_k \hat{x}_k(-)]$
Error covariance update	$P_k(+) = (I - \bar{K}_k H_k) P_k(-)$
Kalman gain matrix	$\bar{K}_k = P_k(-) H_k^T [H_k P_k(-) H_k^T + R_k]^{-1}$

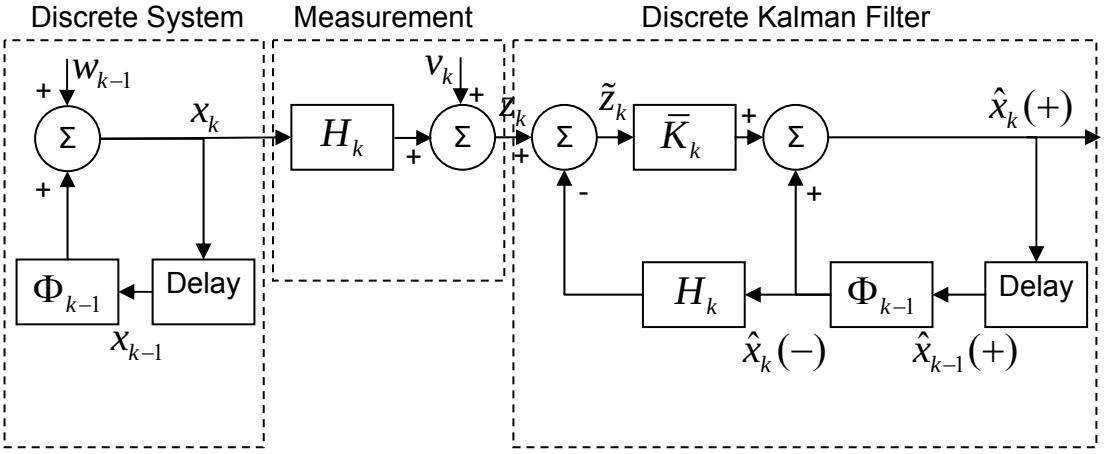


Figure 2.1 Block diagram of system, measurement model, and discrete-time Kalman filter.

CHAPTER THREE

IMAGE DENOISING VIA KALMAN FILTERING USING MULTIPLE FRAMES OF IMAGES

The subject of this chapter is to realize an image denoising algorithm by using one dimensional Kalman filter method applied to images that contain multiple frames. Since the most important part of the problem is the definition of the state space model and Kalman filter equations, it is better to look at how a scalar estimate is realized by using Kalman filter.

3.1 Scalar Estimation by Using One Dimensional Kalman Filter

Kalman filter is a multiple-input, multiple-output digital filter that can optimally estimate, in real time, the states of a system based on its noisy outputs. These states are all the variables needed to completely describe the system behavior as a function of time (such as position, velocity, voltage levels, and so forth). In fact, one can think of the multiple noisy outputs as a multidimensional signal plus noise, with the system states being the desired unknown signals. The Kalman filter then filters the noisy measurements to estimate the desired signals. The estimates are statistically optimal in the sense that they minimize the mean-square estimation error. This has been shown to be a very general criterion in that many other reasonable criteria (the mean of any monotonically increasing, symmetric error function such as the absolute value) would yield the same estimator.

Figure 3.1 illustrates the Kalman filter algorithm itself. Because the state (or signal) is typically a vector of scalar random variables (rather than a single variable), the state uncertainty estimate is a covariance matrix. Each diagonal term of the matrix is the variance of a scalar random variable (a description of its uncertainty). The matrix's off-diagonal terms are the covariances that describe any correlation between pairs of variables.

The multiple measurements (at each time point) are also vectors that a recursive algorithm processes sequentially in time. This means that the algorithm iteratively repeats itself for each new measurement vector, using only values stored from the previous cycle. This procedure distinguishes itself from batch-processing algorithms, which must save all past measurements.

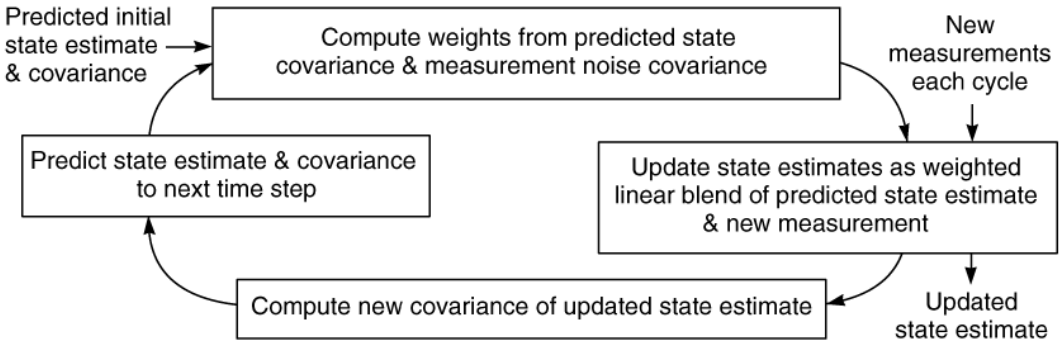


Figure 3.1 The cycle of a recursive Kalman filter.

Starting with an initial predicted state estimate (as shown in Figure 3.1) and its associated covariance obtained from past information, the filter calculates the weights to be used when combining this estimate with the first measurement vector to obtain an updated "best" estimate. If the measurement noise covariance is much smaller than that of the predicted state estimate, the measurement's weight will be high and the predicted state estimate's will be low.

Because the filter calculates an updated state estimate using the new measurement, the state estimate covariance must also be changed to reflect the information just added, resulting in a reduced uncertainty. The updated state estimates and their associated covariances form the Kalman filter outputs.

Finally, to prepare for the next measurement vector, the filter must project the updated state estimate and its associated covariance to the next measurement time. The actual system state vector is assumed to change with time according to a deterministic linear transformation plus an independent random noise. Therefore, the

predicted state estimate follows only the deterministic transformation, because the actual noise value is unknown. The covariance prediction accounts for both, because the random noise's uncertainty is known. Therefore, the prediction uncertainty will increase, as the state estimate prediction cannot account for the added random noise. This last step completes the Kalman filter's cycle.

One can see that as the measurement vectors are recursively processed, the state estimate's uncertainty should generally decrease (if all states are observable) because of the accumulated information from the measurements. However, because information is lost (or uncertainty increases) in the prediction step, the uncertainty will reach a steady state when the amount of uncertainty increase in the prediction step is balanced by the uncertainty decrease in the update step. If no random noise exists in the actual model when the state evolves to the next step, then the uncertainty will eventually approach zero. Because the state estimate uncertainty changes with time, so too will the weights. Generally speaking, the Kalman filter is a digital filter with time-varying gains.

If the state of a system is constant, the Kalman filter reduces to a sequential form of deterministic, classical least squares with a weight matrix equal to the inverse of the measurement noise covariance matrix. In other words, the Kalman filter is essentially a recursive solution of the least-squares problem (Levy, 2002).

A scalar estimation process like a DC voltage estimation or resistor value estimation can be given as a simple example of using Kalman filter. In this case, the scalar value does not change with time, but the measured values of scalar change because of the process and measurement noises.

Then, the linear state space model can be defined as the following equation set

$$x_k = \Phi_{k-1} x_{k-1} + w_{k-1},$$

$$w_k \sim \mathrm{N}\left(0, Q_k\right),$$

$$z_k = H_k x_k + v_k,$$

$v_k \sim \mathrm{N}(0, R_k)$.

where $\Phi_{k-1} = H_k = I$ (I shows the identity matrix), w_k is the process noise, v_k is the measurement noise, and $k = 0,1,\ldots$.

If the unknown parameters and initial conditions are defined properly, the recursive Kalman filter equation set starts to work and the result of the filter becomes closer to the actual output value of the system at each time step.

Let the unknown scalar parameter represent a DC voltage with a 5V value. Also, let the process noise and measurement noise covariances be given as $Q_k = 0.01$ and $R_k = 0.1$. Finally, the initial estimate of the state and the error covariance matrix are determined as $x_{-1} = 0$ and $P_0(+) = 1$.

A simple software algorithm is written with MATLAB to realize such a problem. First of all, the state space model is defined and the noisy measurements are realized by adding process and measurement noises to the actual value. Then, the Kalman filter algorithm is executed and its outputs are compared with the actual value. Figure 3.2 shows the output signal of the Kalman filter.

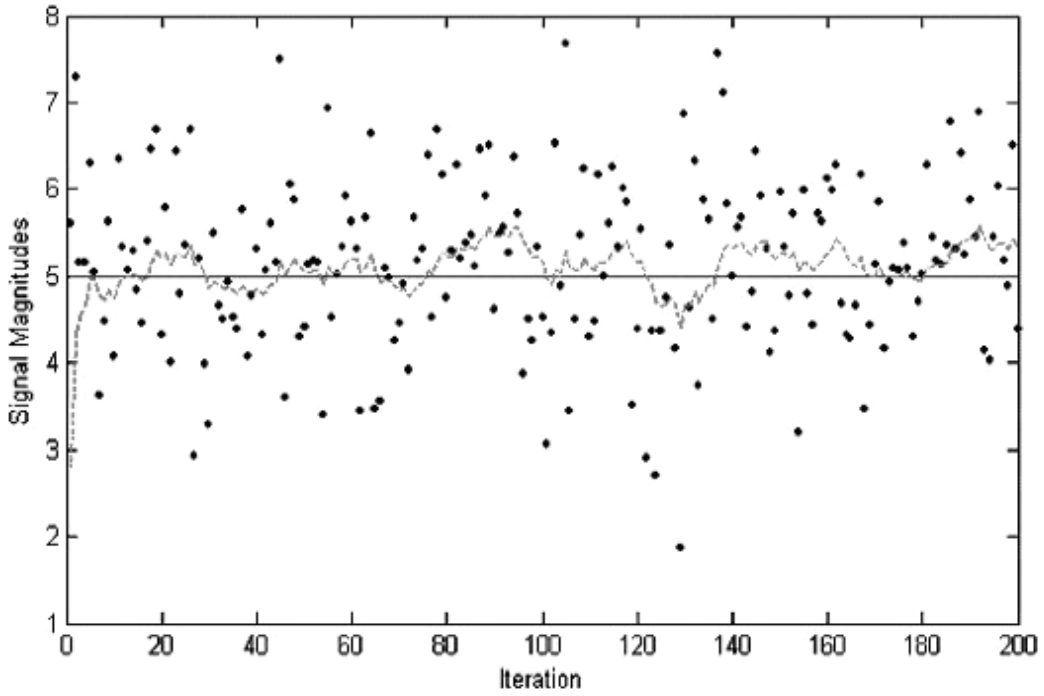


Figure 3.2 The comparison of the actual signal (shown as ——), the measured signal (shown as ···) and the estimated signal (shown as – –).

Figure 3.3 shows the Kalman gain and the error variance with respect to iteration number.

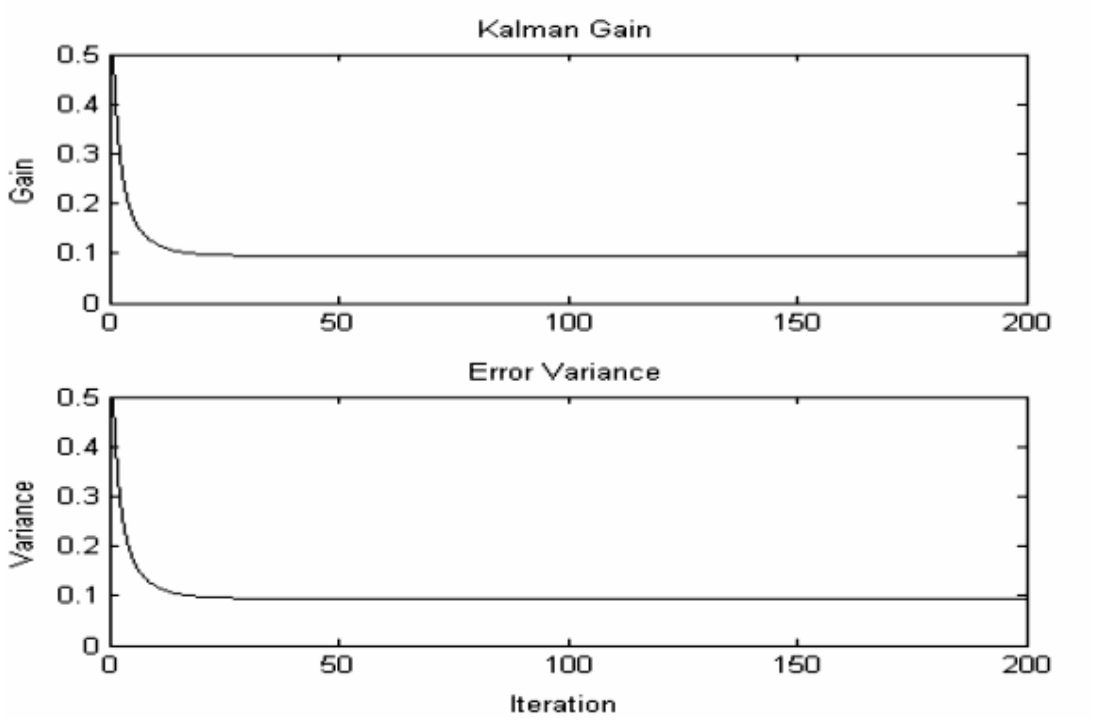


Figure 3.3 Kalman gain and the error variance.

As shown in Figure 3.2, the actual signal is constant and the measured value differs according to the noise covariance parameters. The output of the filter becomes closer to the actual value at about 10th iteration. Then, the error variance and the Kalman gain become stable.

3.2 Denoising of Multiple Frame Image

In this part, the problem of denoising images with multiple frames will be defined and it will be shown how the solution of the problem can be realized by using one dimensional Kalman filter method explained in the previous section.

First of all, the term *"frame"* represents images taken at different times consecutively with different noise realizations. If the content of the image does not change quickly and the images are taken fast enough, then one can get the same image with different noise realizations. As a result, even if the pixel values of the obtained noisy images are not equal to each other, original image is the same. The position of a pixel and the process of taking images to form frames are given in Figure 3.4 and Figure 3.5, respectively.

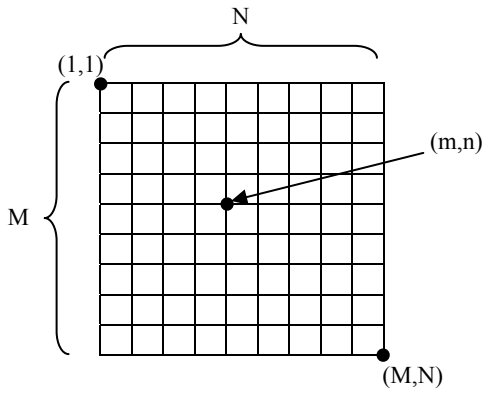


Figure 3.4 The position of a pixel at (m,n). The size of the image is MxN.

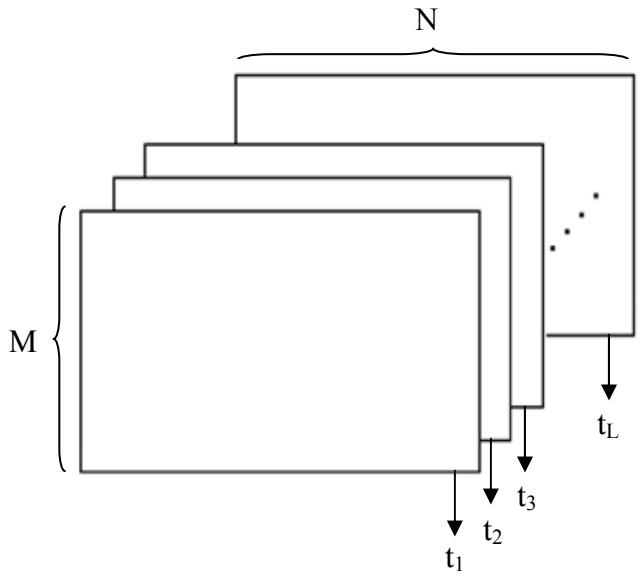


Figure 3.5 The process of forming frames by taking images consecutively.

The process of image taking starts at time t_1 and goes on until time t_L. This means that L frames are taken between times t_1 and t_L.

The important isssue at this point is that how can one dimensional Kalman filter can be applied for an image. Naturally, an image is a two dimensional spatial representation in most applications. But, 2D to 1D conversion should be realized to be able to apply 1D Kalman filter method. For this purpose, various methods are suggested like scanning the image in horizontal and vertical directions. In our case, it is known that the deniosed images should be equal to each other. Therefore, the pixel values of the same spatial positions –at the same horizontal and vertical coordinates- of each frame should have the same values. By using this fact, the following equation set can be written for an image of size MxN;

$$x(m,n,k) = Ax(m,n,k) + Bw(m,n,k), \tag{3.1}$$

$$z(m,n,k) = Cx(m,n,k) + v(m,n,k) \tag{3.2}$$

where $k = 1,2,\ldots,L$, $m = 1,2,\ldots,M$, and $n = 1,2,\ldots,N$. As a result, the problem becomes a scalar estimation of specific coordinates.

If a specific pixel value is wanted to be estimated, the formulation can be rearranged. For example, for $m = a$ and $n = b$ the formulation becomes,

$$x(a,b,k) = Ax(a,b,k) + Bw(a,b,k), \tag{3.3}$$

$$z(a,b,k) = Cx(a,b,k) + v(a,b,k), \tag{3.4}$$

where $1 \le a \le M$ and $1 \le b \le N$.

Since the pixel positions are constant, the following formulation can be written,

$$x(k) = Ax(k) + Bw(k), \tag{3.5}$$

$$z(k) = Cx(k) + v(k). \tag{3.6}$$

Therefore, the problem becomes an estimation of one dimensional constant signal for a specific pixel position along the frames taken at different times.

The algorithm for the system defined above is realized in MATLAB environment. A grayscale *Lena* image is used for the experiments with its range changing between 0 and 1. The value of *'0'* corresponds to *black* and the value of *'1'* corresponds to *white*. The image consists of 256 gray levels. The results are obtained by changing three parameters which are number of frames, process noise and measurement noise. At each experiment, the value of one parameter –number of frames, process noise or measurement noise- is changed and the others are held constant. The results obtained for different number of frames, process and measurement noise values are given in Tables 3.1, 3.2 and 3.3, respectively.

PSNR (Peak Signal to Noise Ratio) is used for performance comparison instead of SNR. PSNR is most easily defined via the mean squared error (MSE). For two $M \times N$ monochrome images I and K, with one of them considered as a noisy approximation of the other, MSE is defined as,

$$MSE = \frac{1}{MN} \sum_{i=0}^{M-1} \sum_{j=0}^{N-1} \| I(i,j) - K(i,j) \|^2, \tag{3.7}$$

Then, PSNR is defined using MSE as,

$$PSNR = 10\log_{10}\left(\frac{MAX_I^2}{MSE}\right) = 20\log_{10}\left(\frac{MAX_I}{\sqrt{MSE}}\right) \tag{3.8}$$

where MAX_I is the maximum possible pixel value of the image. Typical PSNR value range is between 20 and 40 dB.

Table 3.1 PSNR values of the noisy and estimated images obtained for different number of frames.

Number of Frames	**Noisy Image (dB)**	**Estimated Image (dB)**
1	13.894	13.894
2	13.894	16.409
3	13.894	17.983
4	13.894	19.314
5	13.894	20.221
10	13.894	23.079
20	13.894	25.615
30	13.894	26.830
40	13.894	27.439
50	13.894	27.779
75	13.894	28.063
100	13.894	28.136

The results in Table 3.1 are obtained by using fixed process and measurement noise values. Process and measurement noise variances are taken as 0.05 and 0.0001, respectively.

As shown in Table 3.1, PSNR value of the estimated image becomes higher as the number of frames increases. This result is parallel to the explanation of the filter characteristics in Section 3.1; that is *"the result of the filter becomes closer to the actual output value of the system at each time step"*. Using greater number of frames reduces the error covariance and therefore PSNR measurement at the output is

improved with more frames. Accordingly, it can also be said that the error covariance decreases with more number of frames.

On the other hand, there is a trade-off in this algorithm as is the case in most of the sciences. If the used number of frames increases, the processing time of the algorithm will be longer. In this case, it can be said that if the processing time of the algorithm using only one frame takes T times, the processing time of the algorithm using two frames takes $2T$ times.

The output images of the filter with some of the given parameter values in Table 3.1 are shown in Figure 3.6. The resulting images show the correspondence of Table 3.1 and Figure 3.6.

Figure 3.6 (a) Original Image, (b) Noisy Image (PSNR = 13,894 dB), (c) $L = 5$ frames, (d) $L = 10$ frames, (e) $L = 20$ frames, (f) $L = 50$ frames.

As can be seen from Figure 3.6, if the number of frames increases, the output image becomes clearer and noise free for this image.

Table 3.2 PSNR values of the noisy and estimated images obtained for different measurement noise variances.

Measurement Noise Variance	**Noisy Image (dB)**	**Estimated Image (dB)**	**Difference between Noisy and Estimated Image (dB)**
0.2	9.651	19.631	9.980
0.1	11.573	22.563	10.990
0.05	13.869	25.650	11.781
0.025	16.523	28.668	12.145
0.01	20.207	32.206	11.999
0.005	23.062	34.300	11.238
0.0025	25.925	35.842	9.917
0.001	29.650	37.652	8.002

The results in Table 3.2 are obtained by using fixed number of frames and process noise. Number of frames and the process noise variance are defined as 20 and 0.0001, respectively.

In Table 3.2, PSNR values of the filter output are compared with the measurement noise change. It is known that the characteristics of Kalman filter is determined by the value of its Kalman gain. If the noise variance gets larger, the filter output is closer to the estimated value as determined by the Kalman gain. On the contrary, if the noise variance gets smaller, the filter output is closer to the measured values. It can easily be seen that the PSNR of the noisy image gets closer to the PSNR of the image at the filter output. This result shows that the pixel values of the noisy image becomes closer to the pixel values of the image at the output of the filter as the measurement noise variance gets smaller. Therefore, the measured values become more reliable compared to the estimated values. Another important point is that the difference between the noisy and the estimated image gets smaller as the noise

variance takes smaller values. The cause of this result is related with the noise level of the image. If the noise level is low in an image, PSNR improvement of the filtered image takes a value in a smaller range according to an image corrupted with a high noise level.

Table 3.3 PSNR values of the noisy and estimated images obtained for different process noise variances.

Process Noise Variance	**Noisy Image (dB)**	**Estimated Image (dB)**	**Difference between Noisy and Estimated Image (dB)**
0.2	9.060	10.415	1.355
0.1	10.366	12.521	2.155
0.05	11.571	14.823	3.252
0.025	12.528	16.998	4.470
0.01	13.292	19.520	6.228
0.005	13.615	21.133	7.518
0.001	13.872	24.337	10.465
0.0005	13.884	25.117	11.233
0.0001	13.869	25.650	11.781

The results in Table 3.3 are obtained by using fixed number of frames and measurement noise variance. Number of frames and the measurement noise variance are taken as 20 and 0.05, respectively.

The effect of changing the process noise variance on the PSNR value at the output can be observed in Table 3.3. As was the case with the measurement noise, the output PSNR value improves as the process noise decreases for this image.

Some other image restoration techniques can also be used for solving this problem. For example, Wiener filter method and averaging method can be used for restoring the noisy image.

Wiener filter is a noise filter based on Fourier iteration. Its main advantage is the short computational time it takes to find a solution.

Consider a situation such that there is some underlying, uncorrupted signal $u(t)$ that is required to measure. Error occur in the measurement due to imperfection in equipments, thus the output signal is corrupted. There are two ways the signal can be corrupted. First, the equipment can convolve, or 'smear' the signal. This occurs when the equipment doesn't have a perfect, delta function response to the signal. Let $s(t)$ be the smear signal and $r(t)$ be the known response that cause the convolution. Then $s(t)$ is related to $u(t)$ by,

$$s(t) = \int_{-\infty}^{\infty} r(t-\tau)u(\tau)d\tau ,$$

or,

$$S(f) = R(f)U(f) . \tag{3.9}$$

where S , R , U are Fourier transform of s , r , and u .

The second source of signal corruption is the unknown background noise $n(t)$. Therefore, the measured signal $c(t)$ is a sum of $s(t)$ and $n(t)$.

$$c(t) = s(t) + n(t) \tag{3.10}$$

To deconvolve s to find u, simply divide $S(f)$ by $R(f)$, i.e. $U(f) = \frac{S(f)}{R(f)}$ in the absence of noise n. To deconvolve c where n is present then one need to find an optimum filter function $\phi(t)$, or $\Phi(f)$, which filters out the noise and gives a signal $\tilde{u}$ by:

$$\tilde{U}(f) = \frac{C(f)\Phi(f)}{R(f)} \tag{3.11}$$

where $\tilde{u}$ is as close to the original signal as possible.

For $\tilde{u}$ to be similar to *u*, their differences squared is as close to zero as possible, i.e.

$$\int_{-\infty}^{\infty} \left|\tilde{u}(t) - u(t)\right|^2 dt ,$$

or

$$\int_{-\infty}^{\infty} \left|\tilde{U}(f) - U(f)\right|^2 df . \tag{3.12}$$

Substituting equation (3.9), (3.10) and (3.11), the Fourier version becomes,

$$\int_{-\infty}^{\infty} \left|R(f)\right|^{-2} \left|S(f)\right|^{-2} \left|1 - \Phi(f)\right|^{-2} + \left|N(f)\right|^{-2} \left|\Phi(f)\right|^{-2} df . \tag{3.13}$$

The best filter is one where the above integral is a minimum at every value of *f*. This is when,

$$\Phi(f) = \frac{\left|S(f)\right|^2}{\left|S(f)\right|^2 + \left|N(f)\right|^2} \tag{3.14}$$

By using the below equation,

$$\left|S(f)\right|^2 + \left|N(f)\right|^2 \approx \left|C(f)\right|^2 \tag{3.15}$$

where $\left|C(f)\right|^2$, $\left|S(f)\right|^2$ and $\left|N(f)\right|^2$ are the power spectrum of *C*, *S*, and *N*.

Therefore,

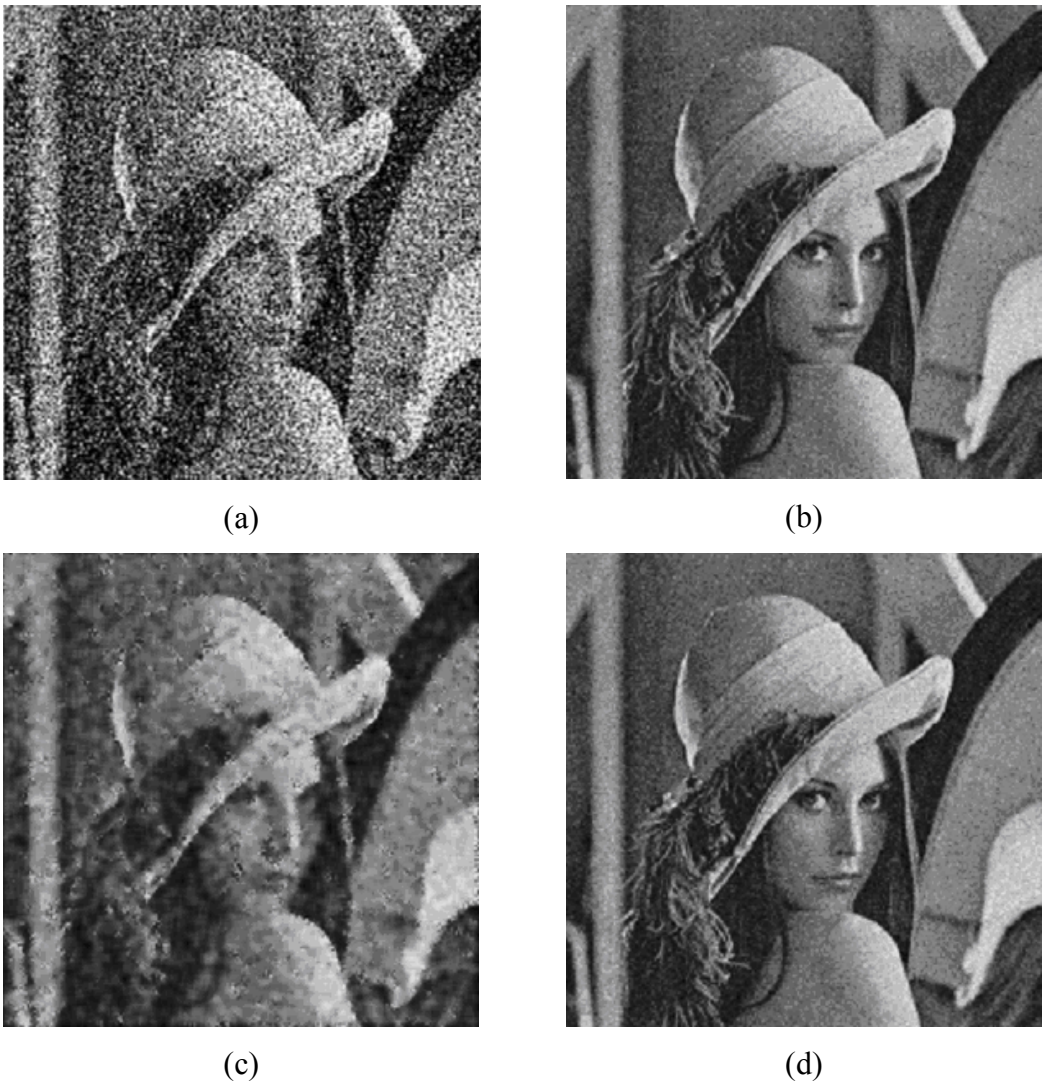
$$\Phi(f) \approx \frac{\left|S(f)\right|^2}{\left|C(f)\right|^2} . \tag{3.16}$$

On the other hand, averaging method is simpler than Wiener method. By considering the formulations between Equation 3.1 and Equation 3.6, averaging method can be formulized as,

$$\hat{x}(m,n) = \frac{1}{L} \sum_{k=1}^{L} x(m,n,k) \tag{3.17}$$

where $\hat{x}(m,n)$ is the estimated image.

Some experiments are performed with Wiener filter and the averaging methods and their results are tabulated in Table 3.4 and compared with the results of Kalman filter method in Figure 3.7.



(a) (b)

(c) (d)

Figure 3.7 (a) Noisy Image (PSNR = 13,894 dB), (b) Kalman filtered image (L=20 & PSNR = 25.615), (c) Wiener filtered image (PSNR = 21.848), (d) Averaging method (L=20 & PSNR = 25.738).

The results in Figure 3.7 are obtained by using 20 frames, and for measurement noise variance of 0.05 and process noise variance of 0.0001.

It can be observed from Figure 3.7 that different image restoration methods produce different output images. However, the results show that Kalman filtering and

averaging methods give similar output images and similar PSNR values for *Lena* image. On the other hand, even if the PSNR value of the Wiener method seems close to the other methods, its output image quality is quite low visually, compared to the Kalman filtering and averaging methods (see Figure 3.7).

Table 3.4 PSNR values of the noisy and estimated images using different image restoration techniques.

Number of Frame	**Noisy Image (dB)**	**Kalman Estimated Image (dB)**	**Wiener Estimated Image (dB)**	**Estimated Image with Averaging Method (dB)**
1	13.894	13.894	21.848	13.894
2	13.894	16.409	21.848	16.858
3	13.894	17.983	21.848	18.515
4	13.894	19.314	21.848	19.747
5	13.894	20.221	21.848	20.591
10	13.894	23.079	21.848	23.372
20	13.894	25.615	21.848	25.738
30	13.894	26.830	21.848	27.011
40	13.894	27.439	21.848	27.804
50	13.894	27.779	21.848	28.347
75	13.894	28.063	21.848	29.270
100	13.894	28.136	21.848	29.789

The results in Table 3.4 are obtained by using fixed process and measurement noises. Process and measurement noise variances are taken as 0.0001 and 0.05, respectively.

As can be seen from Table 3.4, all of the methods have their distinct advantages within some parameter value ranges for *Lena* image. If the number of frames is less than or equal to 5, then the best way for the restoration is to use Wiener filter method. On the other hand, if the number of frames is greater than 5, then Kalman filter and averaging methods give better results.

Another important point is the processing times. The processing times of the algorithms are given in Table 3.5.

Table 3.5 Processing times of Kalman filter, Wiener filter and averaging methods.

Number of Frame	**Kalman Estimated Image (seconds)**	**Wiener Estimated Image (seconds)**	**Estimated Image with Averaging Method (seconds)**
1	2	0.031	0.0003
2	4	0.031	0.0006
3	6	0.031	0.0010
4	9	0.031	0.0013
5	11	0.031	0.0016
10	21	0.031	0.0032
20	41	0.031	0.0064
30	62	0.031	0.0096
40	83	0.031	0.0128
50	104	0.031	0.0160
75	155	0.031	0.0240
100	207	0.031	0.0320

If three methods are compared according to their processing times, it can be said that averaging method has the shortest processing time and Kalman filter method has the longest processing time. Naturally, the number of frames affects the processing times of Kalman filter and the averaging methods because, more operations are performed with more number of frames. Since Wiener filter always uses only one frame, the frame number does not affect its processing time.

CHAPTER FOUR

A FULL-PLANE BLOCK KALMAN FILTER FOR IMAGE RESTORATION

In this chapter, a 2-D block Kalman filtering method is used for image restoration that is based on (Citrin & Azimi-Sadjadi, 1992). For this purpose, an algorithm is written in MATLAB. Simulation results, output figures and processing times are given at the end of this chapter.

4.1 Usage of 2-D Kalman Filtering in Image Restoration

Although the concepts of classical filter theory and its extensions have been successfully exploited, the recursive estimation techniques (Kalman filters) have only recently been applied in image processing (Nahi & Assefi, 1972), (Habibi, 1972), (Jain, 1977), (Woods & Radewan, 1977), (Hart, 1975), (Aboutalib, 1977). Most of the reported techniques using Kalman algorithms have either considered the degradation due to random noise only (no blurring), thus simplifying the computing requirements, or degradation due to both blurring and random noise but requiring extensive computing power due to the very long system state vectors needed to account for blurring.

(Nahi & Assefi, 1972), in their pioneering work, demonstrated the feasibility of applying Kalman algorithms to restore noisy images by using the scalar scanner output as the output of a dynamic system whose input consisted of white noise. Such a system model was developed by transforming the planar brightness distribution to a form suitable for use in Kalman one-step predictor algorithms. (Habibi, 1972) has suggested a two-dimensional recursive filter which gives a Bayesian estimate of an image from a two-dimensional noise observation, eliminating the non-stationary effects due to the scanner approach of (Nahi & Assefi, 1972). (Jain, 1977) developed a semi-causal representation of images and deduced scalar recursive filtering equations (first order Markov processes), thus reducing the number of computations required to estimate images degraded by white additive noise. In an interesting paper

by (Woods & Radewan, 1977), a Kalman vector processor has been used in strips in order to reduce the computational load. The resulting filter, called a strip processor, is based on the assumption that correlation in an image decreases substantially at large distances.

The use of a reduced update filter (which is scalar) as an approximation to a Kalman vector processor has also been suggested and shown to be optimum in that it minimizes the post update mean-square error under the constraint of updating only the nearby previously processed neighbors (Dikshit, 1982). Although above methods in employing Kalman filters are encouraging, they are limited in application to the noisy images only.

To maintain the proper state dynamics within the state and the error covariance equations (Sage & Melsa, 1971) and to design an optimal Kalman filter, a large state vector and correspondingly large error covariance matrices would be involved. This, obviously, leads to an excessively large amount of storage and computations. A number of researchers introduced various filtering schemes (Woods & Ingle, 1981), (Suresh & Shenoi, 1981), (Azimi-Sadjadi, Bannour, & Citrin, 1989), (Azimi-Sadjadi, & Bannour, 1991) to overcome these problems.

The idea of the reduced update Kalman filtering (RUKF) (Suresh & Shenoi, 1981), (Azimi-Sadjadi, Bannour, & Citrin, 1989), (Azimi-Sadjadi, & Bannour, 1991) is to partition the state vector into two segments; the “local state” and the “global state.” The “local state” propagates in both dimensions during the filtering process and consists of a group of pixels, in the region of support of the model, which are spatially close to the points being estimated. The “global state,” however, contains those previously estimated pixels needed to estimate the future ones. Substantial computational saving is achieved by using only the local state in the filtering process and by transferring the data from the global state into the local state whenever the initial estimate is generated. Pixels which are not included in the local state tend to be less correlated and provide little information for generating the initial estimate due to the Markovian assumption.

The region of support of the 2-D model can have various geometry and types, namely, causal, semi-causal (or half-plane), and non-causal (or full plane) (Ranganath & Jain, 1985). Although non-causal models are shown (Ranganath & Jain, 1985) to provide better match to the actual image correlations, causal support was widely used in almost all of the previous methods. However, when a causal model is used more than half of the pixels in the adjoining support are ignored in generating the initial estimate. Therefore, it is desirable to devise a way to generate a full-plane model for more accurate estimation while maintaining the causality within the filtering process.

4.2 Two-Dimensional State-Space Modelling and Full-Plane Block Kalman Filter Definitions

Consider an image of size MxM which is scanned vectorially from left to right and top to bottom in block rows of width N_1. The image is assumed to be represented by a zero-mean vector Markov process. Each block within a block row is of size $N = N_1 x N_2$, where N_1 is the number of rows of pixels within a block, N_2 is the number of columns in a block, and N is the number of pixels in each block. A block row is defined as a strip of blocks extending across the image from left to right and is of size $N_1 xM$. A block row contains M / N_2 blocks, assuming that M is divisible by N_2. The process is illustrated in Figure 4.1. The pixels within a block are arranged in row ordered form. A processing strip, hereafter called a strip, consists of three block rows. The goal is to estimate the blocks in the middle block row. The upper and lower block rows consist of block estimates generated to provide support for the blocks in the middle block row. Two estimates of the blocks in the middle block row need to be generated. The first estimates are generated solely to provide support to the second estimator. The second block estimates in the middle block row will be saved as the final filtered estimate.

The local state-space model for the image process is given by

$$\begin{bmatrix} X_0(k) \\ X_1(k) \\ . \\ . \\ . \\ X_8(k) \end{bmatrix} = A \begin{bmatrix} X_0(k-1) \\ X_1(k-1) \\ . \\ . \\ . \\ X_8(k-1) \end{bmatrix} + \begin{bmatrix} U_0(k) \\ U_1(k) \\ . \\ . \\ . \\ U_8(k) \end{bmatrix} \tag{4.1a}$$

or

$$X(k) = AX(k-1) + BU(k) \tag{4.1b}$$

where $X(k)$ is the current state vector consisting of 9 blocks $X_i(k)$, $i \in [0,8]$; $X(k-1)$ is the past state vector consisting of 9 blocks $X_i(k-1), i \in [0,8]$, and $U(k)$ is a zero mean white driving noise vector process and is of size $(9xN)x1$. The first $5xN$ elements of $U(k)$ are equal to zero.

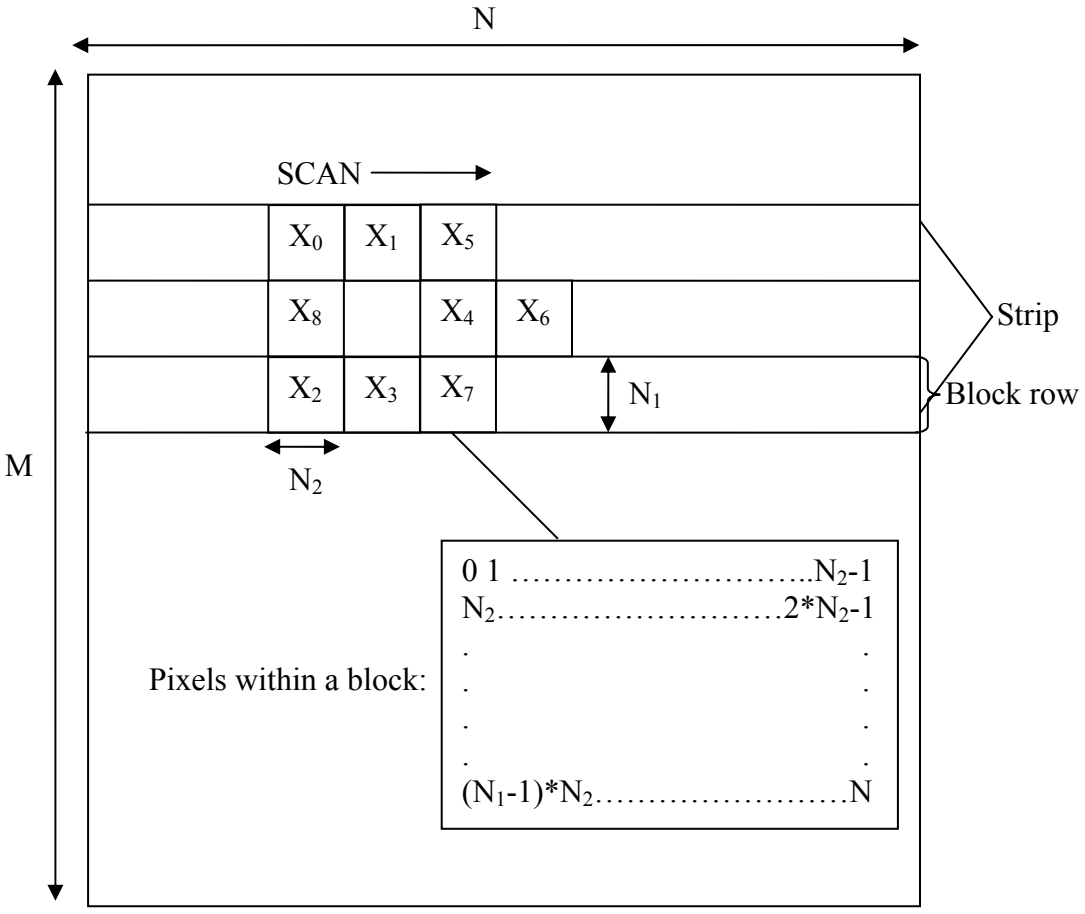


Figure 4.1 Size of blocks. Block row, state, and numbering of pixels within a block.

The spatial positions of $X_i(k)$ at a given iteration "k" are shown in Figure 4.1. The peculiar numbering of these blocks is solely chosen to provide easier programming by getting all the blocks which are to be estimated, i.e., blocks 5, 6, 7, and 8, numerically close together and their support numerically adjacent. The blocks which are not filtered estimates are obtained by shifting the blocks within the state as the state advances to the right, so that the previously estimated blocks occupy the proper spatial positions within the state. Figure 4.2 illustrates the state propagation along horizontal direction with each iteration.

The supports for blocks 5, 6, 7, and 8 are given in Table 4.1;

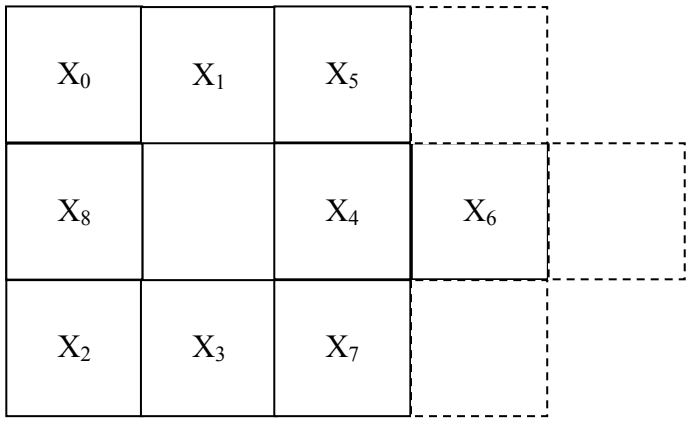
Table 4.1 Support blocks corresponding to the filtered blocks.

Filtered Block	**Support**
$X_5(k)$	$X_i(k-1),\ i=4,5,6,7$
$X_6(k)$	$X_i(k-1),\ i=6$
$X_7(k)$	$X_i(k-1),\ i=4,5,6,7$
$X_8(k)$	$X_i(k-1),\ i=0,1,2,3,4,5,6,7,8$

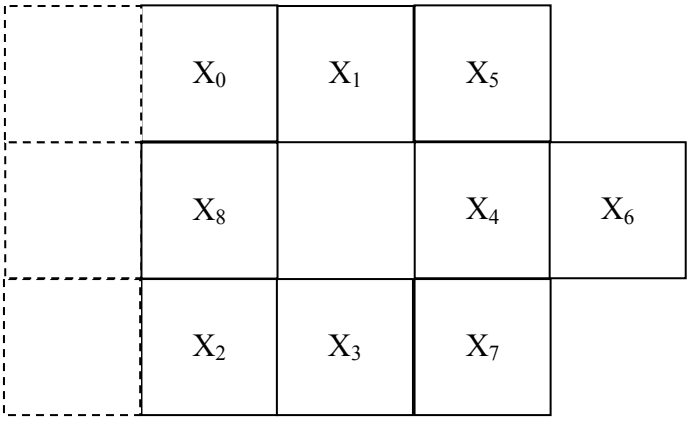
This results in an $\boldsymbol{A}$ matrix of size $(9xN)x(9xN)$ which is given by,

$$A=\begin{bmatrix} 0 & I & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & I & 0 & 0 & 0 \\ 0 & 0 & 0 & I & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & I & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & I & 0 & 0 \\ 0 & 0 & 0 & 0 & A_{54} & A_{55} & A_{56} & A_{57} & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & A_{66} & 0 & 0 \\ 0 & 0 & 0 & 0 & A_{74} & A_{75} & A_{76} & A_{77} & 0 \\ A_{80} & A_{81} & A_{82} & A_{83} & A_{84} & A_{85} & A_{86} & A_{87} & A_{88} \end{bmatrix} \tag{4.2}$$

Each sub-matrix of A is of size NxN **.**



$X(k-1)$



$X(k)$

Figure 4.2 Propagation of the state $X_0(k)$ and $X_1(k-1)$ occupy the same spatial position.

The procedure in each strip begins by advancing the strip one block row down. The propagation of the state along the boundaries will be discussed later. The state propagates along the strip from left to right and as it advances Blocks 0, 1, 2, 3, and 4 are shifted from the previous state and Blocks 5, 6, 7, and 8 are estimated using four concurrent estimators. Block 5 is re-estimated to avoid the storage of this block from the previous block row, which could have resulted in large error covariance matrices. Block 7 is an intermediate estimate of data that will again be estimated as Blocks 6 and 8 when the strip is advanced to the next block row. Block 6, which is ahead of Block 8, is estimated based upon the past Block 6 in the same block row to provide

the right side support of Block 8. The filtered estimate of Block 8 with a full-plane of support is the only block estimate which is saved.

As can be seen, by using these multiple concurrent block estimators causality is maintained within the actual filtering process. Furthermore, re-estimating those blocks along the upper and lower block rows avoids the need to store large error covariances associated with these states, thus resulting in substantial reduction in computation efforts and storage requirements. The only disadvantage is that the recursion occurs in one direction.

4.3 Parameter Estimation for the Image Modelling

In this section a procedure is given to obtain the model parameters for the four estimators ($X_i(k), i = 5,6,7,8$) described in Section 4.2. The model parameters to be estimated are A_i, i.e., the ith block row of the A matrix and the correlation matrices $Q_{U_{ij}} \triangleq E[U_i(k)U_j^T(k)]$, $i, j = 0,1,...,8$, where $E[\cdot]$ is the expectation operator. These are the diagonal sub-matrices of the covariance matrix, Q_U, of the driving process, $U(k)$.

Let us begin by considering the first block estimator $X_6(k)$ which has only one support block. To obtain the model parameters A_6, and $Q_{U_{66}}$, for this block estimator we extract the corresponding row of (4.1) i.e.;

$$X_6(k) = A_6 X(k-1) + U_6(k). \tag{4.3.a}$$

Ignoring the zero portion of A_6, we simply obtain

$$X_6(k) = A_{66} X_6(k-1) + U_6(k) \tag{4.3b}$$

where $X_6(k)$ and $U_6(k)$ are of size $Nx1$ and A_{66}, which is the seventh sub-matrix of A_6, is of size NxN .

By post-multiplying by $X_6^T(k-1)$, taking the expectation, and using the orthogonality principle (Sage & Melsa, 1971);

$$E[U_6(k)U_6^T(k)] = 0 \tag{4.4}$$

yields

$$\rho_{66}(l) = A_{66}\rho_{66}(l-1) + Q_{U_{66}}\delta(l) \tag{4.5a}$$

where

$$\rho_{ij}(l) \triangleq E[X_i(n)X_j^T(n-l)]. \tag{4.5b}$$

Transposing (4.5a) and using the property $\rho_{ij}^T(l) = \rho_{ij}(-l)$ gives,

$$\rho_{66}(-l) = \rho_{66}(1-l)A_{66}^T + Q_{U_{66}}\delta(l) \tag{4.6}$$

which is the normal equation for this estimator. Plugging $l = 0,1$ in this equation gives the following vector Yule-Walker equation,

$$\begin{bmatrix} \rho_{66}(0) & \rho_{66}(1) \\ \rho_{66}^T(1) & \rho_{66}(0) \end{bmatrix}\begin{bmatrix} I \\ -A_{66}^T \end{bmatrix} = \begin{bmatrix} Q_{U_{66}} \\ 0 \end{bmatrix} \tag{4.7}$$

which can be rearranged and solved to give $Q_{U_{66}}$ and A_{66}.

To obtain the parameters A_5 and $Q_{U_{55}}$ for the second block estimator $X_5(k)$, and A_7 and $Q_{U_{77}}$ for the third estimator $X_7(k)$, the same procedure can be repeated. For example, for the second estimator the state equation is,

$$X_5(k) = A_5X(k-1) + U_5(k) \tag{4.8}$$

which is obtained by extracting the sixth block row of (4.1). Alternatively, we have

$$X_5(k) = \begin{bmatrix} A_{54} & A_{55} & A_{56} & A_{57} \end{bmatrix}\begin{bmatrix} X_4(k-1) \\ X_5(k-1) \\ X_6(k-1) \\ X_7(k-1) \end{bmatrix} + U_5(k). \tag{4.9}$$

Using a similar procedure and invoking the orthogonality principle gives the relevant vector Yule-Walker equation as,

$$\begin{bmatrix} \rho_{55}(0) & \rho_{54}(1) & \rho_{55}(1) & \rho_{56}(1) & \rho_{57}(1) \\ \rho_{54}^{T}(1) & \rho_{44}(0) & \rho_{45}(0) & \rho_{46}(0) & \rho_{47}(0) \\ \rho_{55}^{T}(1) & \rho_{54}(0) & \rho_{55}(0) & \rho_{56}(0) & \rho_{57}(0) \\ \rho_{56}^{1}(1) & \rho_{64}(0) & \rho_{65}(0) & \rho_{66}(0) & \rho_{67}(0) \\ \rho_{57}^{T}(1) & \rho_{74}(0) & \rho_{75}(0) & \rho_{76}(0) & \rho_{77}(0) \end{bmatrix} \begin{bmatrix} I \\ -A_{54}^{T} \\ -A_{55}^{T} \\ -A_{56}^{T} \\ -A_{57}^{T} \end{bmatrix} = \begin{bmatrix} Q_{U_{55}} \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}. \tag{4.10}$$

The solution of this system of equations provides $Q_{U_{55}}$ and $A_{5i}, i = 4,5,6,7$. For the third estimator, $X_7(k)$, we simply replace $X_5(k)$ with $X_7(k)$, $U_5(k)$ with $U_7(k)$, A_{5i} with A_{7i} and ρ_{5i} with ρ_{7i} ($i = 4,5,6,7$) in (4.9) and (4.10).

For the estimation of the model parameters for the last block estimator, $X_8(k)$, we form

$$\begin{bmatrix} \rho_{88}(0) & \rho_{80}(1) & \rho_{81}(1) & \rho_{82}(1) & \rho_{83}(1) & \rho_{84}(1) & \rho_{85}(1) & \rho_{86}(1) & \rho_{87}(1) & \rho_{88}(1) \\ \rho_{80}^{T}(1) & \rho_{00}(0) & \rho_{01}(0) & \rho_{02}(0) & \rho_{03}(0) & \rho_{04}(0) & \rho_{05}(0) & \rho_{06}(0) & \rho_{07}(0) & \rho_{08}(0) \\ \rho_{81}^{T}(1) & \rho_{10}(0) & \rho_{11}(0) & \rho_{12}(0) & \rho_{13}(0) & \rho_{14}(0) & \rho_{15}(0) & \rho_{16}(0) & \rho_{17}(0) & \rho_{18}(0) \\ \rho_{82}^{T}(1) & \rho_{20}(0) & \rho_{21}(0) & \rho_{22}(0) & \rho_{23}(0) & \rho_{24}(0) & \rho_{25}(0) & \rho_{26}(0) & \rho_{27}(0) & \rho_{28}(0) \\ \rho_{83}^{T}(1) & \rho_{30}(0) & \rho_{31}(0) & \rho_{32}(0) & \rho_{33}(0) & \rho_{34}(0) & \rho_{35}(0) & \rho_{36}(0) & \rho_{37}(0) & \rho_{38}(0) \\ \rho_{84}^{T}(1) & \rho_{40}(0) & \rho_{41}(0) & \rho_{42}(0) & \rho_{43}(0) & \rho_{44}(0) & \rho_{45}(0) & \rho_{46}(0) & \rho_{47}(0) & \rho_{48}(0) \\ \rho_{85}^{T}(1) & \rho_{50}(0) & \rho_{51}(0) & \rho_{52}(0) & \rho_{53}(0) & \rho_{54}(0) & \rho_{55}(0) & \rho_{56}(0) & \rho_{57}(0) & \rho_{58}(0) \\ \rho_{86}^{T}(1) & \rho_{60}(0) & \rho_{61}(0) & \rho_{62}(0) & \rho_{63}(0) & \rho_{64}(0) & \rho_{65}(0) & \rho_{66}(0) & \rho_{67}(0) & \rho_{68}(0) \\ \rho_{87}^{T}(1) & \rho_{70}(0) & \rho_{71}(0) & \rho_{72}(0) & \rho_{73}(0) & \rho_{74}(0) & \rho_{75}(0) & \rho_{76}(0) & \rho_{77}(0) & \rho_{78}(0) \\ \rho_{88}^{T}(1) & \rho_{80}(0) & \rho_{81}(0) & \rho_{82}(0) & \rho_{83}(0) & \rho_{84}(0) & \rho_{85}(0) & \rho_{86}(0) & \rho_{87}(0) & \rho_{88}(0) \end{bmatrix} \begin{bmatrix} I \\ -A_{80}^{T} \\ -A_{81}^{T} \\ -A_{82}^{T} \\ -A_{83}^{T} \\ -A_{84}^{T} \\ -A_{85}^{T} \\ -A_{86}^{T} \\ -A_{87}^{T} \\ -A_{88}^{T} \end{bmatrix} = \begin{bmatrix} Q_{U_{88}} \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix} \tag{4.11}$$

which can be solved to give $Q_{U_{88}}$ and $A_{8i}, i \in [0,8]$.

So far we have obtained all the required sub-matrices of matrix A and also identified the diagonal sub-matrices of Q_U, $Q_{U_{ij}}$, $i = j$. In order to complete the modeling process, the off-diagonal sub-matrices $Q_{U_{ij}}$, $i \neq j$, have to be determined. For the $Q_{U_{56}}$ matrix, post-multiply both sides of (4.9) by $X_6^T(k)$ and use the orthogonality principle, $E[U_5(k)X_6^T(k-1)] = 0$ which gives,

$$Q_{U_{56}} = \rho_{56}(0) - \begin{bmatrix} A_{54} & A_{55} & A_{56} & A_{57} \end{bmatrix} \begin{bmatrix} \rho_{46}(-1) \\ \rho_{56}(-1) \\ \rho_{66}(-1) \\ \rho_{76}(-1) \end{bmatrix}, \tag{4.12a}$$

and similarly, for $Q_{U_{76}}$ and $Q_{U_{57}}$. For $Q_{U_{58}}$ we obtain,

$$Q_{U_{58}} = \rho_{58}(0) - \begin{bmatrix} \rho_{50}(1) & \rho_{51}(1) & \rho_{52}(1) & \rho_{53}(1) & \rho_{54}(1) & \rho_{55}(1) & \rho_{56}(1) \\ \rho_{57}(1) & \rho_{58}(1) \end{bmatrix} A_8, \tag{4.12b}$$

and similarly, for $Q_{U_{78}}$ and $Q_{U_{68}}$. The remaining off-diagonal sub-matrices can be obtained by the relationship, $Q_{U_{ji}} = Q_{U_{ij}}^T$. For, $i \in [0,4], j \in [0,8]$ and $i \in [5,8], j \in [0,4]$ $Q_{U_{ij}} = 0$, i.e., zero sub-matrix as the corresponding blocks are obtained solely by the shifting process.

4.4 Kalman Filtering Process

Once all the model parameters are identified, the block dynamic (4.1) can be formed. The observation equation in block form is,

$$Z(k) = HX(k) + V(k) \tag{4.13}$$

where $Z(k)$ is the corrupted image or observation vector of size $(4xN)x1$, $V(k)$ is the observation noise vector of the same size as $Z(k)$ containing a scalar zero mean white Gaussian additive noise $v(k)$ with variance σ_v^2, and H is a $(4xN)x(9xN)$ matrix containing the elements of the point spread function (PSF) of the non-causal blur.

For the case of an image corrupted solely by additive noise H becomes,

$$H = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & I & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & I & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & I & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & I \end{bmatrix} \tag{4.14}$$

where 0 represents a zero sub-matrix each of size NxN. The Kalman filter equations for the system in (4.1) and (4.13) are,

$$P_b(k) = AP_a(k-1)A^T + BQ_U B^T \tag{4.15a}$$

$$K(k) = P_b(k)H^T(HP_b(k)H^T + Q_V)^{-1} \tag{4.15b}$$

$$\hat{X}(k) = A\hat{\hat{X}}(k-1) \tag{4.15c}$$

$$\hat{\hat{X}}(k) = \hat{X}(k) + K(k)(Z(k) - H\hat{X}(k)) \tag{4.15d}$$

$$P_a(k) = [I - K(k)H]P_b(k) \tag{4.15e}$$

where $\hat{X}(k)$ is the a priori (before updating) estimate, $\hat{\hat{X}}(k)$ is the a posteriori (after updating) estimate, $P_b(k)$ is the a priori error covariance matrix defined by,

$$P_b(k) \triangleq E[(X(k) - \hat{X}(k))(X(k) - \hat{X}(k))^T], \tag{4.16a}$$

$P_a(k)$ is the a posteriori error covariance matrix defined by,

$$P_a(k) \triangleq E[(X(k) - \hat{\hat{X}}(k))(X(k) - \hat{\hat{X}}(k))^T], \tag{4.16b}$$

$K(k)$ is the Kalman gain matrix and Q_U, and Q_V, are correlation matrices of the independent processes U and V, respectively. The state is reinitialized at the beginning of each strip. With the exception of the boundaries of the image, only the estimate $\hat{\hat{X}}_8(k)$ is saved as a final estimate. After a strip is processed, one advances a block row and starts a new strip without using any filtered estimates from the prior strip. Thus recursion only occurs along a strip.

4.5 Boundary Conditions

There are four boundary conditions to be considered, which correspond to top, bottom, left, and right edges of the image. The condition at the beginning of a strip (left boundary) is related to the state at iteration $k = -1$. The blocks that are outside the image, i.e., $X_i(-1)$, $i = 0,2,8$ are initialized to the mean of the image while those inside the image, i.e., $X_i(-1), i = 1,3,4,5,6,7$, are initialized to the observed noisy data corresponding to those in spatial position.

At the beginning of each strip one also needs to initialize $P_a(-1)$ matrix. For those estimates spatially located outside the image, the diagonal elements of $P_a(-1)$ are chosen as the variance of the image since the mean is used as the estimate. For those estimates using the observation, the diagonal elements are the variance of the noise, since this corresponds to the squared error associated with these estimates. The off diagonal elements of $P_a(-1)$ are expected to be zero.

Next, we consider the boundary conditions at the right edge of the image, i.e., at the end of a strip. As the state approaches the right side boundary of the image at iteration $k = (M / N_2 - 1) - 3$ the image is processed like the previous iterations, with block $X_8(k)$ being saved as a final filtered estimate. For the three subsequent and final iterations in the strip, both $X_6(k)$ and $X_8(k)$ are saved as final estimates for their proper spatial positions. This ensures that all spatial positions of the final filtered image contain filtered data without having to contend with the fact that no observation exists outside the right boundary of the image.

For the first strip of the image located spatially within the image, estimate $X_5(k)$ is saved as the final filtered estimate for the first block row and estimate $X_8(k)$ is saved as a final filtered estimate for the second block row. The first two blocks, and also the last block of the first block row are left unfiltered, i.e., the observation is used as the final estimate. The last block row is processed the same as the first block row except that $X_7(k)$ is saved as the final filtered estimate. The first two and last blocks of this block row are also left unfiltered.

4.6 Simulation Results

In this section, implementation of the algorithm and simulation results will be given along with some tables and figures. The experiments have been repeated for different block sizes. In this work, two different block sizes have been used. One of

them has been chosen as 1x1 and the other as 2x2. Also, the simulations results have been obtained for different measurement noise variance values and repeated many times. Since the process noise covariance matrix is calculated as part of the algorithm, we are not allowed to change its value. Finally, the algorithm has been tried for different images to be able to observe the performance of the filter for different inputs.

In tables, both the PSNR and SNR results have been shown. SNR values have been calculated according to the formula shown below,

$$SNR = 10\log_{10}\left(\frac{\sigma_S^2}{\sigma_N^2}\right) \tag{4.17}$$

where σ_S^2 and σ_N^2 shows the variance of the image and the noise, respectively.

First of all, measurement noise variance has been chosen as a fixed value and a grayscale *Lena* image is used for the simulations. The range of the pixel values of the *Lena* image is between 0 and 1. The value of *'0'* corresponds to *black* and the value of *'1'* corresponds to *white*.

Firstly, measurement noise variance value has been chosen as 0.01 and 1x1 block size has been used. The results are shown in Figure 4.3 and Table 4.2.

(a) (b)

(c)

Figure 4.3 (a) Original Lena image, (b) Noisy image (PSNR = 20.269dB, SNR = 6.5669), (c) Filtered image (Block size = 1x1).

Table 4.2 Results of the simulation of *Lena. T*he measurement noise variance is 0.01 and 1x1 block size is used.

	Original image	**Noisy image**	**Filtered image**
Mean of the image	0.3831	0.3855	0.3831
Variance of the image	0.042635	0.009399	0.0029607
SNR(dB)	∞	6.5669	11.5777 (+5,0108)
PSNR(dB)	∞	20.2690	25.2798 (+5,0108)

To be able to see the changing of the Kalman gain and a posteriori error, Kalman gain and a posteriori error values of “Block 8” are plotted for a random row in the

image (here it is selected as row 100). The results are shown in Figure 4.4 and Figure 4.5.

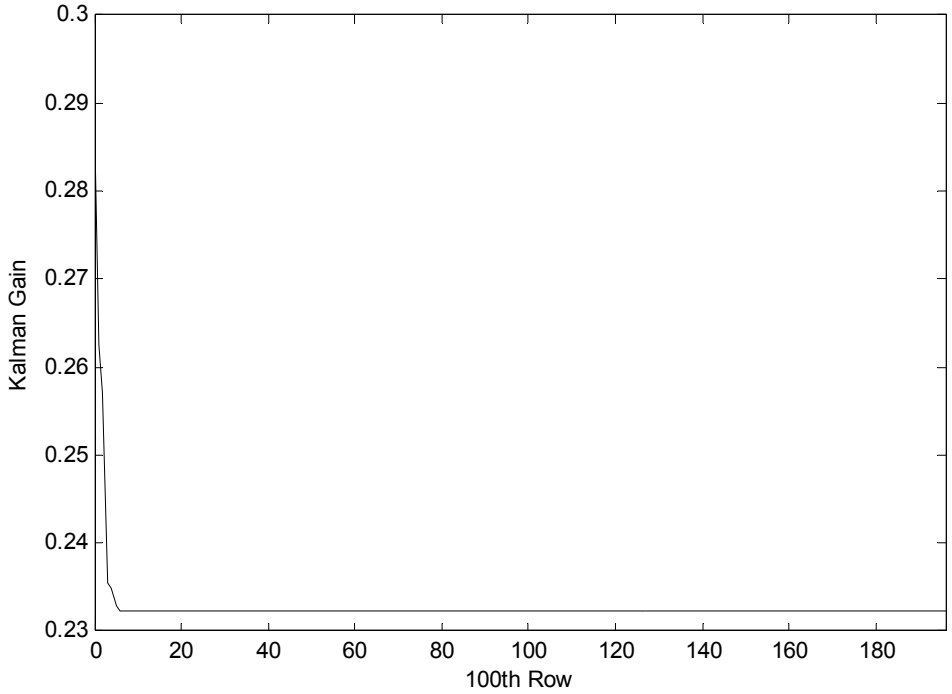


Figure 4.4 Kalman gain of "Block 8" for the 100th row of the image (Block size = 1x1).

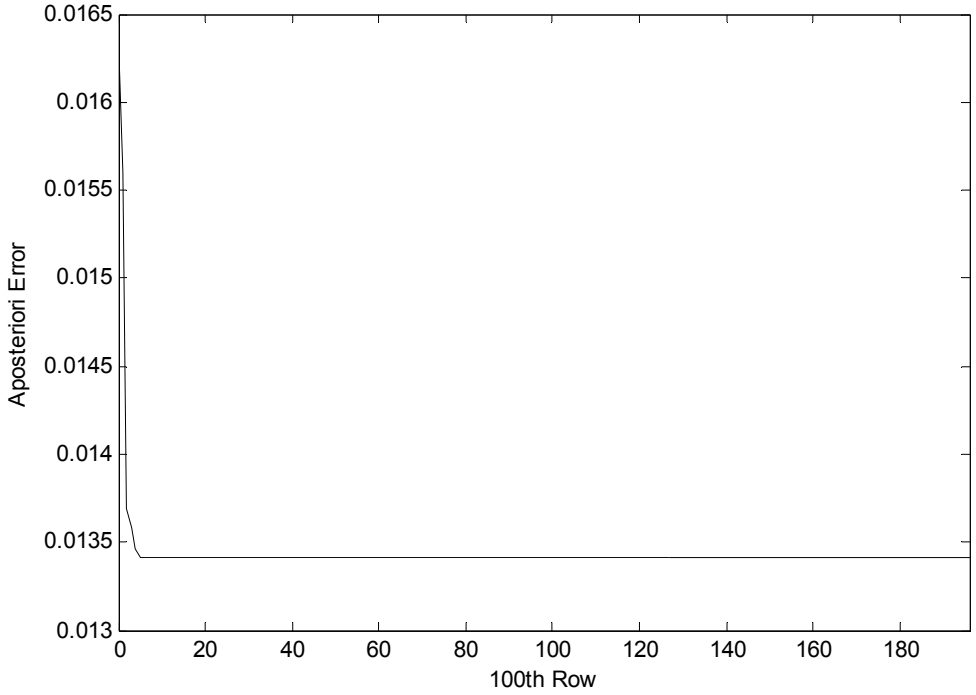


Figure 4.5 Aposteriori error of "Block 8" for the 100th row of the image (Block size = 1x1).

Results in Figure 4.4 and Figure 4.5 are similar with the results in Figure 3.3 in the previous section. Therefore, parallel comments can be made for Figure 4.4 and Figure 4.5. Kalman gain and a posteriori error values get smaller at the beginning and they stay constant after almost 10 iterations.

Secondly, the same algorithm is used with 2x2 block size. The other parameters of the algorithm are the same with the previous experiment. The results of this experiment are shown in Figure 4.6 and Table 4.3.

(a) (b)

Figure 4.6 (a) Noisy image (PSNR = 20.269, SNR = 6.5668), (b) Filtered image (Block size = 2x2).

Table 4.3 Results of the simulation of *Lena*. *T*he measurement noise variance is 0.01 and 2x2 block size is used.

	Noisy image	**Filtered image**
SNR(dB)	6.5668	11.159 (+4,5922)
PSNR(dB)	20.269	24.861 (+4,5922)

As can be seen from Figure 4.6 and Table 4.3, the filtered *Lena* image looks better visually compared to the noisy images. The resulting image is smoother than the noisy image and its PSNR and SNR values are better. If the effect of the block size is considered, it can be said that 1x1 block choice gives better results compared to the 2x2 block.

On the other hand, experiments are also performed with different measurement noise variances. In the previous experiments, the value of the measurement noise variance was selected as 0.01. Now, the results of the experiments will be given for different noise variance values that are changing between 0.001 and 0.1. At first, the results of 1x1 block size are summarized in Table 4.4.

Table 4.4 Results of the simulation of *Lena*. *T*he measurement noise variance is changing between 0.001 and 0.1 and 1x1 block size is used.

Measurement Noise Variance		**Noisy image**	**Filtered image**	**Difference (dB)**
0.001	PSNR(dB)	30.000	31.165	+0.165
	SNR(dB)	16.298	17.463	
0.0025	PSNR(dB)	26.085	28.755	+2.671
	SNR(dB)	12.382	15.053	
0.005	PSNR(dB)	23.115	26.965	+3.850
	SNR(dB)	9.413	13.262	
0.0075	PSNR(dB)	21.469	25.975	+4.506
	SNR(dB)	7.767	12.273	
0.01	PSNR(dB)	20,248	25,259	+5,011
	SNR(dB)	6,546	11,557	
0.025	PSNR(dB)	16.504	22.955	+6.451
	SNR(dB)	2.802	9.253	
0.05	PSNR(dB)	13.889	20.951	+7.062
	SNR(dB)	0.188	7.249	
0.075	PSNR(dB)	12.523	19.792	+7.269
	SNR(dB)	-1.179	6.0895	
0.1	PSNR(dB)	11.503	18.787	+7.284
	SNR(dB)	-2.1991	5.0845	

As can be seen from Table 4.4, the image at the filter output has always better PSNR and SNR values for *Lena* image. If the noise variance gets larger, then the

filter changes its output adaptively and estimated pixel values become more important. However, if the noise variance has a small value, the filter outputs rely more on to the input image pixel values and the output image becomes closer to the input image. As a result, PSNR and SNR differences between the input and output images become less.

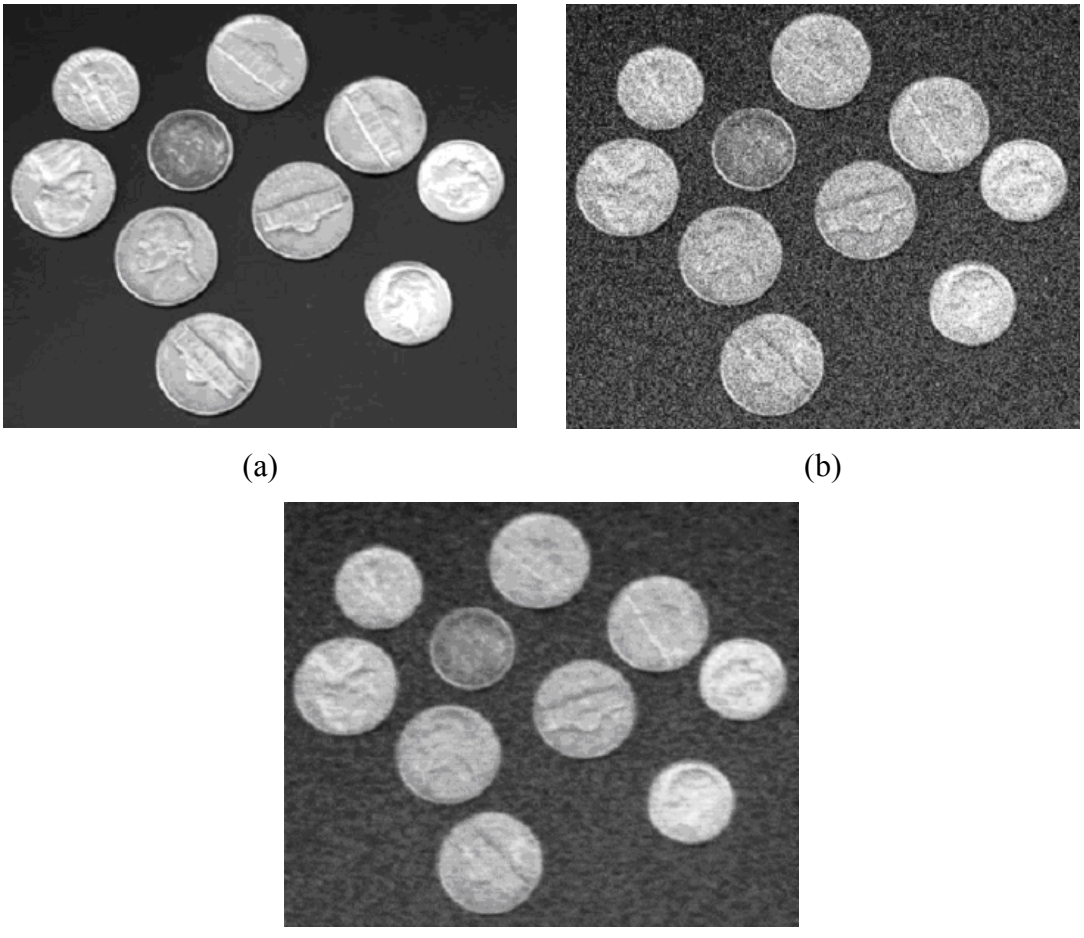
Similar experiments are also done with 2x2 block size and the results are shown in Table 4.5.

Table 4.5 Results of the simulation of *Lena*. *T*he measurement noise variance is changing between 0.001 and 0.1 and 2x2 block size is used.

Measurement Noise Variance		**Noisy image**	**Filtered image**	**Difference (dB)**
0.001	PSNR(dB)	30.042	26.880	-3.162
	SNR(dB)	16.340	13.178	
0.0025	PSNR(dB)	26.072	26.782	+0.710
	SNR(dB)	12.369	13.08	
0.005	PSNR(dB)	23.186	26.277	+3.091
	SNR(dB)	9.484	12.574	
0.0075	PSNR(dB)	21.487	25.592	+4.105
	SNR(dB)	7.785	11.889	
0.01	PSNR(dB)	20.269	24.861	+4,592
	SNR(dB)	6.567	11.159	
0.025	PSNR(dB)	16.469	22.354	+5.885
	SNR(dB)	2.766	8.652	
0.05	PSNR(dB)	13.855	20.456	+6.602
	SNR(dB)	0.153	6.754	
0.075	PSNR(dB)	12.457	19.457	+7.000
	SNR(dB)	-1.245	5.755	
0.1	PSNR(dB)	11.549	18.828	+7.279
	SNR(dB)	-2.154	5.126	

As can be observed from Table 4.5, the filter output gives similar results compared with the filter of 1x1 block size. However, if the noise variance gets smaller, the SNR difference between the filtered and noisy images assumes a negative value (meaning output is worse than input) more quickly than the filter of 1x1 block size.

Finally, the experiment was performed with a different grayscale image named *Coins* to see the consistency of the algorithm. Like the previous experiments, measurement noise variance is selected 0.01 and 1x1 block size is used firstly. The results are shown in Figure 4.7.



(a) (b)

(c)

Figure 4.7 (a) Original image, (b) Noisy image (PSNR = 20.109dB, SNR = 6.9548), (c) Filtered image (Block size = 1x1).

In this image, there are more constant valued regions compared to *Lena* image. Therefore, performance of the filter becomes better. Since the filter output makes the input image smoother, it is expected that PSNR and SNR values should be higher for an image with more constant valued regions. PSNR and SNR values of the output images for different noise variances are tabulated in Table 4.6 and Table 4.7. Results for 1x1 and 2x2 block sizes are given in Table 4.6 and Table 4.7, respectively. Measurement noise variance values are changing between 0.001 and 0.1.

Table 4.6 Results of the simulation of *Coins*. The measurement noise variance is changing between 0.001 and 0.1 and 1x1 block size is used.

Measurement Noise Variance		**Noisy image**	**Filtered image**	**Difference (dB)**
0.001	PSNR(dB)	29.980	30.924	+0.943
	SNR(dB)	16.826	17.770	
0.0025	PSNR(dB)	26.026	30.590	+4.564
	SNR(dB)	12.872	17.436	
0.005	PSNR(dB)	23.034	28.896	+5.8623
	SNR(dB)	9.880	15.742	
0.0075	PSNR(dB)	21.329	27.852	+6.523
	SNR(dB)	8.175	14.698	
0.01	PSNR(dB)	20.070	27.102	+7.032
	SNR(dB)	6.916	13.948	
0.025	PSNR(dB)	16.439	24.589	+8.149
	SNR(dB)	3.285	11.435	
0.05	PSNR(dB)	13.901	22.386	+8.486
	SNR(dB)	0.747	9.232	
0.075	PSNR(dB)	12.563	21.092	+8.530
	SNR(dB)	-0.591	7.939	
0.1	PSNR(dB)	11.626	20.161	+8.535
	SNR(dB)	-1.528	7.007	

As can be seen from the tables, the filter has a better performance compared to the *Lena* image. It can be said that if the content of an image consists of more details, the performance of the filter becomes worse. The reason for this can be explained as removal of the details by the filter during the de-noising process.

At the next table, the same experiment is realized by using 2x2 blocks and all of the other parameter values are kept the same with the previous experiment.

Table 4.7 Results of the simulation of *Coins*. The measurement noise variance is changing between 0.001 and 0.1 and 2x2 block size is used.

Measurement Noise Variance		**Noisy image**	**Filtered image**	**Difference (dB)**
0.001	PSNR(dB)	30.042	26.818	-3.224
	SNR(dB)	16.888	13.664	
0.0025	PSNR(dB)	26.093	27.073	+0.981
	SNR(dB)	12.939	13.919	
0.005	PSNR(dB)	23.049	26.865	+3.817
	SNR(dB)	9.895	13.712	
0.0075	PSNR(dB)	21.305	26.271	+4.967
	SNR(dB)	8.151	13.117	
0.01	PSNR(dB)	20.104	25.973	+5.869
	SNR(dB)	6.951	12.819	
0.025	PSNR(dB)	16.437	23.795	+7.359
	SNR(dB)	3.283	10.641	
0.05	PSNR(dB)	13.976	22.129	+8.153
	SNR(dB)	0.822	8.975	
0.075	PSNR(dB)	12.524	20.987	+8.463
	SNR(dB)	-0.630	7.833	
0.1	PSNR(dB)	11.630	20.259	+8.629
	SNR(dB)	-1.524	7.1047	

Up to this point, noise removal performance of the algorithm is considered. Another important point to look at is the processing time of the algorithm. In the real world, this issue is much more important to be able to use the filter in a real-time application. Therefore, the processing times of the algorithm for 1x1 and 2x2 block sizes are calculated. Naturally, process time of the algorithm is parallel with how many processes are executed. Accordingly, the processing time can be affected only by the size of the image and blocks used for scanning.

Up to now, two images named *Lena* and *Coins* are used for two different experiments. Those two example images have different sizes. As a result, they have different processing times. The size of *Lena* image is 200x200 and the size of *Coins* image is 245x297. The processing times for these two images are given for 1x1 and 2x2 block sizes in Table 4.8.

Table 4.8 Processing times of Lena and Coins images for 1x1 and 2x2 block sizes.

	Processing time for Lena image (seconds)	**Processing time for Coins image (seconds)**
1x1 Block	26.4	71
2x2 Block	11.2	26

The results in Table 4.8 have been obtained via an algorithm written in MATLAB environment. In this experiment, a computer with Intel(R) Core(TM)2 CPU T7200 @ 2.0GHz processor and 1.00GB of RAM was used.

From Table 4.8, it can easily be seen that the larger image requires more processing time. For the same block sizes, the process time of the experiment using *Lena* image takes a shorter time compared to the *Coins* image. On the other hand, block size affects the processing times. For the same image, performing the same experiment with a 2x2 block size requires a shorter time compared to 1x1 block size. This result shows that larger size block operations shorten the processing time.

However, if PSNR and SNR results are taken into consideration, it is seen that large size block operations produce worse performance results. As a result, there is a trade-off between the processing times and SNR improvement of the filtered image when the block size is considered.

CHAPTER FIVE

CONCLUSION

In this thesis, Kalman filtering techniques have been used for the purpose of image restoration. 1-D and 2-D Kalman filtering methods are used for two different scenarios defined in Chapter 3 and Chapter 4. Even if it is hard to apply the Kalman filter to images because of the difficulty of modeling the image as a linear dynamic system and the excessive processing time, it has still been a worthy experience.

In the first experiment, 1-D Kalman filter is used for de-noising of an image. A different scenario is defined for this problem. It is assumed that the same image is obtained within a short time period consecutively and saved using different noise realizations. Then, the resulting images with the same content and different noise realizations are de-noised with a 1-D Kalman filter. According to these assumptions, a model is constructed and the original image is attempted to be estimated by applying 1-D Kalman filter to each pixel of the image. In this work, the number of obtained images determines the iteration number. Therefore, more images mean more number of iterations and better results. It is shown that Kalman filter gives better results compared to the Wiener filtering method after a specific iteration number. However, the processing time of the algorithm is excessive. Even if the results of the averaging and Kalman filtering methods are close to each other, averaging method has a clear advantage because of its simplicity and short processing time. It is seen that, 1-D Kalman filtering can be applied in this scenario. However, it is not very practical to use especially when averaging method is considered.

In the second experiment, de-noising of images is performed with a 2-D Kalman filtering method. This time a more complex modeling is realized by closely following the research paper titled *"A Full-Plane Block Kalman Filter for Image Restoration"* written by S. Citrin and M. R. A. Sadjadi as a reference. According to this work, a 2-D Kalman filtering method is presented which uses a full-plane image

model to generate a more accurate filtered estimate of an image that has been corrupted by additive noise and full-plane blur. Estimation of the original image is realized by using block processing. One block is estimated at each iteration by using supported blocks defined near the estimated block. The model provides causality within the filtering process by employing multiple concurrent block estimators. After modeling and simulation steps of the work are finished, performance of the filter is observed by changing different parameters. Firstly, the output of the filter is observed for different block sizes and then for different measurement noise variances. Finally, the performance of the filter is observed with a different example image to judge the consistency of the filter.

It is seen that the filter performance changes for different block sizes. Smaller block sizes give better results. On the other hand, the experiments performed with smaller block sizes lengthen the processing time of the algorithm. Moreover, if the noise variance becomes larger, the filter adaptively changes its output and estimation process of the Kalman filter greatly affects the output. On the contrary, if the noise variance gets smaller, the effect of the estimation process decreases. Finally, experiments performed with different images show that the algorithm works suitably with various images. However, it is seen that images with more regions of constant value produce better results compared to more detailed images.

REFERENCES

Aboutalib A. O. (June 1977). Digital restoration of images degraded by general motion blurs. *IEEE Trans. Automat. Contr.*, vol. AC-22, pp. 294-302.

Azimi-Sadjadi M. R., Bannour S., & Citrin S. (May 1989). A 2-D adaptive diagonal block Kalman filter for nonsymmetric half plane image models. *Proc. IEEE Int. Symp. on Circuits and Systems,* Portland, OR, pp. 1528-1531.

Azimi-Sadjadi M. R., & Bannour S. (September 1991). Two dimensional adaptive block Kalman filtering of SAR imagery. *IEEE Trans. Geosci. Remote Sensing,* vol. 29, pp. 742-753.

Citrin S., & Azimi-Sadjadi M. R. (October 1992). A full-plane block Kalman filter for image restoration. *IEEE Transactions on Image Processing,* vol. 1(4), pp. 488-495.

Dikshit S. S. (April 1982). A recursive window approach to image restoration. *IEEE Transactions on Acoustics, Speech and Signal Processing,* ASSP-30(2), pp. 125-140.

Grewal M. S., & Andrews A. P. (2001). *Kalman Filtering: Theory and Practice Using MATLAB* (2nd ed.). John Wiley & Sons, Inc.

Habibi A. (July 1972). Two-dimensional Bayesian estimate of images. *Proc. IEEE,* vol. 60, pp. 878-883.

Hart C. *G.* (1975). Digital signal processing for image de-convolution and enhancement. *New Directions in Signal Processing in Communication and Control, J. K. Skwirzynski, Ed., NATO ASIS. Groningen, The Netherlands: Noordhoff-Leyden.*

Jain, A. K. (1988). *Fundamentals of Digital Image Processing*. Englewood Cliffs, NJ: Prectice Hall.

Jain A. K. (April 1977). A semicausal model for recursive filtering of two dimensional images. *IEEE Trans. Comput.,* vol. C-26, pp. 343-350.

Levy L. J. (2002). *The Kalman Filter: Navigation's Integration Workhorse*, The Johns Hopkins University Applied Physics Laboratory.

Maybeck, P. S. (1979). *Stochastic Models, Estimation, and Control*. New York, NY: Academic Press.

Nahi N. E., & Assefi T. (July 1972). Bayesian recursive image estimation. *IEEE Trans. Comput.,* vol. C-21, pp, 734-738.

Ranganath S., & Jain A. K. (February 1985). Two-dimensional linear prediction models-part I: Spectral factorization and realization. *IEEE Trans. Acoust., Speech Signal Processing,* vol. ASSP 33, pp. 280-299.

Sage A. P., & Melsa J. L. (1971). *Estimation Theory with Applications to Communications and Control.* New York, NY: McGraw-Hill.

Suresh B. R., & Shenoi B. A. (April 1981). New results in two-dimensional Kalman filtering with applications to image restorations. *IEEE Trans. Circuits Syst.,* vol. CAS-28, pp. 307-319.

Woods J. W., & Radewan C. H. (July 1977). Kalman filtering in two dimensions. *IEEE Trans. Inform. Theory,* vol. IT-23, pp. 473-482.

Woods J. W., & Ingle V. K. (April 1981). Kalman filtering in two-dimensions: Further results. *IEEE Trans. Acoust., Speech Signal Processing,* vol. ASSP-29, pp. 188-197.