NONLINEAR IMAGE RESTORATION

A THESIS SUBMITTED TO
THE GRADUATE SCHOOL OF NATURAL AND APPLIED SCIENCES
OF
MIDDLE EAST TECHNICAL UNIVERSITY

BY

CAHİT UĞUR UNGAN

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR
THE DEGREE OF MASTER OF SCIENCE
IN
ELECTRICAL AND ELECTRONICS ENGINEERING

NOVEMBER 2005

Approval of the Graduate School of Natural and Applied Sciences

_____

Prof. Dr. Canan ÖZGEN
Director

I certify that this thesis satisfies all the requirements as a thesis for the degree of Master of Science.

_____

Prof. Dr. İsmet ERKMEN
Head of Department

This is to certify that we have read this thesis and that in our opinion it is fully adequate, in scope and quality, as a thesis for the degree of Master of Science.

_____

Prof. Dr. Kerim DEMİRBAŞ
Supervisor

Examining Committee Members

Prof. Dr. Kemal LEBLEBİCİOĞLU (METU, EE) _____

Prof. Dr. Kerim DEMİRBAŞ (METU, EE) _____

Assoc. Prof. Dr. Bilgehan GÜVEN (METU, STAT) _____

Assoc. Prof. Dr. Fahrettin ARSLAN (Ankara University, STAT) _____

Asist. Prof. Dr. Çağatay CANDAN (METU, EE) _____

I hereby declare that all information in this document has been obtained and presented in accordance with academic rules and ethical conduct. I also declare that, as required by these rules and conduct, I have fully cited and referenced all material and results that are not original to this work.

Name, Last name : Cahit Uğur UNGAN

Signature :

ABSTRACT

NONLINEAR IMAGE RESTORATION

UNGAN, Cahit Uğur
M.S., Department of Electrical and Electronics Engineering
Supervisor : Prof. Dr. Kerim DEMİRBAŞ

November 2005, 166 pages

This thesis analyzes the process of deblurring of degraded images generated by space-variant nonlinear image systems with Gaussian observation noise. The restoration of blurred images is performed by using two methods; a modified version of the Optimum Decoding Based Smoothing Algorithm [1] and the Bootstrap Filter Algorithm [18] which is a version of Particle Filtering methods. A computer software called MATLAB is used for performing the simulations of image estimation. The results of some simulations for various observation and image models are presented.

Keywords: Space-Variant Nonlinear Image Systems, Optimum Decoding Based Smoothing Algorithm, Bootstrap Filter Algorithm, Particle Filtering Methods

ÖZ

DOĞRUSAL OLMAYAN RESİM DÜZELTMESİ

UNGAN, Cahit Uğur
Yüksek Lisans, Elektrik ve Elektronik Mühendisliği Bölümü
Tez Yöneticisi : Prof. Dr. Kerim DEMİRBAŞ

Kasım 2005, 166 sayfa

Bu çalışma, uzayda değişen doğrusal olmayan, Gaussian gözlem gürültüsü içeren görüntü oluşturma sistemleri tarafından bozulmuş görüntülerin bulanıklığının giderilmesi sürecini incelemiştir. Bulanık görüntülerin onarımı, Optimum Kod Çözümüne dayalı Düzeltme Algoritmasının [1] değiştirilmiş bir uyarlaması ve Parçacık Süzgeçleme yöntemlerinin bir uyarlaması olan *Bootstrap* Süzgeç Algoritması [18] olmak üzere iki yöntem kullanılarak yapılmıştır. Resim tahmini simülasyonlarını gerçekleştirmek için MATLAB isimli bir bilgisayar yazılımı kullanılmıştır. Çeşitli gözlem ve görüntü modelleri için bazı simülasyon sonuçları sunulmuştur.

Anahtar Kelimeler: Uzayda Değişen Doğrusal Olmayan Görüntü Sistemleri, Optimum Kod Çözümüne Dayalı Düzeltme Algoritması, Bootstrap Süzgeç Algoritması, Parçacık Süzgeçleme Yöntemleri

To Ayşe, Sedef & Ömer

ACKNOWLEDGMENTS

I would like to thank my supervisor Prof. Dr. Kerim Demirbaş for his guidance, advice, criticism, encouragements and insight throughout this thesis work.

I would also like to thank Volkan Emre Arpınar for his precious tips and tricks about MATLAB.

I would like to express my deepest gratitude to my parents, Sedef and Ömer Ungan for their encouragement and support throughout my life in all aspects. I would also like to show my appreciation to my grandmother, Sevinç Tuncer for her invaluable support for my education life as a wise teacher and for her blessings as a loving grandmother. I would like to thank Gülay Taşpınar for the spark she created in a desperate moment. I am also grateful for all my relatives and friends for making me who I am.

Additionally, I would also like to extend my gratitudes to my colleagues at Aselsan Inc. for their support, guidance and valuable advice throughout my thesis work. I am grateful to Aselsan Inc. for the facilities provided for the completion of this thesis.

Last but not least, thank you Ayşe, for just being with me…

TABLE OF CONTENTS

CHAPTER 1

INTRODUCTION

Image restoration intends to achieve an improvement of the visual quality of degraded image data [7]. It may be viewed as an estimation process in which operations are performed on an observed or measured field to estimate the ideal image field that would be observed if no image degradation were present in an imaging system [4]. It finds application in many different areas like medical thermography, satellite imagery and military target screening [7].

The observed or recorded image is a degraded version of the original or ideal image in various practical cases. The reasons for the image degradation are blurring that affects the image in a deterministic manner and noise which is a stochastic phenomenon that corrupts the image. The main reasons for blurring are atmospheric turbulence, relative motion between camera and object, lens variations and defocused systems. Noise, usually arises due to errors in transmission and noise inherent in the electronics of the image formation system [4].

The main image restoration approaches that have appeared in the literature are classified primarily on the image model assumed, that is, deterministic or stochastic, stationary or nonstationary, dynamic or static, but also on the algorithmic procedure used for obtaining the solution [5]. Traditional image restoration methods are based on linear processing of images; however in most practical image restoration applications, the image formation model is nonlinear [2]. Both photochemical and photoelectric systems involve a built-in nonlinearity. The nonlinearity is introduced in the transformation of light intensity to output units of the imaging system; granular concentration in the film, or current intensity in photoelectric systems [3],

[10], [11]. The nonlinear additive noise model has been established as a generalized descriptor of optical systems [3], [11]-[13]. Many other systems, which couple the signal and the noise in an exponential and/or multiplicative fashion, can be brought to this form through model transformations [11], [14]-[16].

In this thesis, simulations on restorations of blurred images degraded by nonlinear space-variant imaging systems with Gaussian observation noise are performed. Images are created by some nonlinear dynamic state transition equations with Gaussian disturbance noise and after the blurring process, the original images are estimated by using two methods. The first method is a modified version of the Optimum Decoding-Based Smoothing Algorithm (ODSA) [1] and the second one is the Bootstrap Filter Algorithm [18] which is one of the Particle Filtering Methods.

The thesis is organized as follows:

In chapter 2, the fundamentals of imaging systems are given. Then, the concept of image restoration is explained and the observation model given in [2] is reviewed for nonlinear image restoration in chapter 3. In chapters 4 and 5, the two methods used for image restoration are explained. The first method called Optimum Decoding-Based Smoothing Algorithm [1] is given in chapter 4 and the second method called Bootstrap Filter Algorithm [18] is given in the next chapter. Simulations performed to observe the performances of the algorithms on nonlinear image restoration are given in chapter 6. Finally, the thesis is concluded in chapter 7. In the Appendix, approximation of an absolutely continuous random vector by a discrete random vector, which is needed to derive the formulations for ODSA [1], is explained.

CHAPTER 2

IMAGE FORMATION SYSTEMS

Every visual scene is an image, specifically, the image formed by the human eye upon the retina. The eye is, of course, not the only image formation system. Images are formed by optical methods and by penetrating nuclear radiation [8].

The principal elements of an image formation system can be shown in Figure 2-1. The “black box” in Figure 2-1 generates the image by acting upon a radiant energy component of the object. Thus image formation is inherently involved with ascertaining the state of a remote region. In optical systems, the radiant light intensity reflected or emitted by the object is transformed by a set of lenses and apertures [8]. In a radiographic image formation system, the nuclear radiation passing through an object is passed through radiation-opaque apertures and/or pinholes [9].

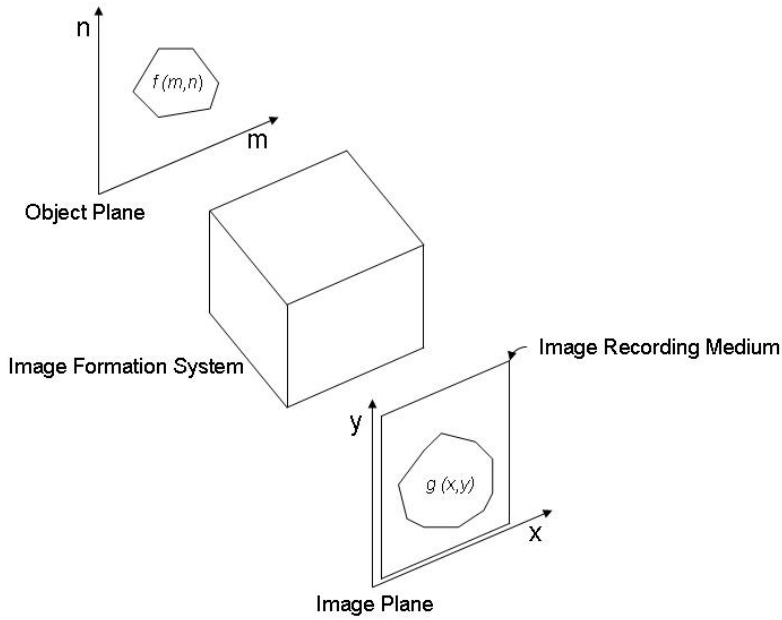

Figure 2-1. Schematic of Image Formation System

The physical form of the devices in the black box of Figure 2-1 is less important than the equations which describe the transformation process [8].

Fortunately, the great variety of image formation processes can be described with a small number of equivalent physical concepts and an associated set of equations. Describing the processes and equations of image formation is motivated by the desire to provide image restoration. Awareness of the processes by which an image was formed before undertaking improvement in the image by restoration is fundamental [3].

2.1. Image Formation [3]

As seen in Figure 2-1 there is assumed to be an object, f(m, n), in the coordinate system (m, n), that is referred to as the object plane. Either the object is

illuminated by a source of radiant energy or the object is itself a source of radiant energy. The radiant energy reflected, transmitted, or emitted by the object propagates through space. An image formation system, the rectangular box in Figure 2-1, intercepts the propagating radiant energy and transforms it in such a manner that in the coordinate system (x, y), which is referred to as the image plane, an image g(x, y) is formed. It should be evident that images are representations of objects that are indirectly sensed and that various forms of radiant energy transport are the mechanisms by which the sensing is carried out.

There are three general principles upon which image formation is based, and the embodiment of these principles in mathematics is required as a point of origin.

2.1.1. Neighborhood Processes

Consider a point (m, n), in the object plane, and the image of that point (x, y), in the image plane, which is created by the image formation system. The image formation system creates the image point (x, y) by acting upon the radiant energy propagating from the object. However, the image formation system receives radiant energy components not only from the object point (m, n) but from all other points in the object. It is logical to expect that as the distance from the object (m, n) to other points in the object plane increases, the effect of these other points on forming the image point will decrease. We must be prepared, however, to recognize the image formation process to be a neighborhood process; i.e., the image of an object point may be dependent on the object point and on points in a (possibly infinite) neighborhood surrounding the object point. If the neighborhood contains only the object point, it is said to be a point process of image formation.

2.1.2. Nonnegativity

The image is formed by the transport of radiant energy. Radiant energy is assumed to propagate from object to image, and the smallest possible amount of radiant energy transport is zero. Thus, the radiant energy distributions for both object and image must be positive or zero; i.e.,

$$f(m, n) \geq 0, \qquad (2.1)$$

$$g(x, y) \geq 0.$$

2.1.3. Superposition

Consider two points in the object plane and the associated radiant energy distributions: $f_1(m, n)$ and $f_2(m, n)$. If only point 1 was emitting radiant energy, then the energy at the image plane may be measured as the quantity $g_1(x, y)$. Likewise, if only point 2 was emitting energy, we would measure energy $g_2(x, y)$ in the image plane. Since the image formation system is responsible for the distribution of energy in the image plane, let us postulate a function that describes the transformation of energy from object plane to image plane. The function must be referenced to both coordinates (m, n) and (x, y), to account for the change in distribution of energy in the different planes. Calling this function h, we describe the energy in the image plane in terms of h and the object radiant energy distribution, f as:

$$g(x, y) = h(x, y, m, n, f(m, n)). \qquad (2.2)$$

The superposition of energy distributions from two points is represented in terms of this function. It is the nature of many transport processes that radiant energy is additive. Therefore, in both object and image plane the radiant energy distributions add; however, the behavior of the image formation system need not relate additive components in the object plane to additive components in the image plane. Such a system is said to be nonlinear, and

$$g_1(x, y) + g_2(x, y) = h(x, y, m, n, f_1(m, n)) + h(x, y, m, n, f_2(m, n))$$

$$g_1(x, y) + g_2(x, y) \neq h(x, y, m, n, f_1(m, n) + f_2(m, n)). \qquad (2.3)$$

If equality holds in the equation (2.3), the image formation system is said to be linear and obeys the principle of superposition under addition. It is trivial to show that if

$$h(x, y, m, n, f(m, n)) = h(x, y, m, n) f(m, n), \tag{2.4}$$

then the image formation system is linear.

Whether linear or nonlinear image formation, the additivity of radiant energy distributions at the image plane makes it possible to describe the image formation process by extending from a single point to an object made up of a continuum. Summing the infinitesimal contributions in the image plane due to all object point contributions in the object plane gives the general image formation equation:

$$g(x, y) = \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} h(x, y, m, n, f(m, n)) dm dn . \tag{2.5}$$

The function h is termed the point-spread function (PSF) of the image formation system. It is evident that it determines the radiant energy distribution in the image plane due to a point source of radiant energy located in the object plane.

The description of the PSF with all four coordinate variables (x, y, m, n) is the most general possible description. Since it allows the point-spread function to vary with the position in both image and object plane, the description in (2.5) is of a space-variant point-spread function (SVPSF).

CHAPTER 3

NONLINEAR IMAGE RESTORATION

The goal of image restoration is to modify a degraded image to produce the original undegraded image. The use of image restoration techniques requires knowledge about how an image was degraded. In other words, a model for the degradation must exist and be known [6].

In imaging systems there are a large number of sources of degradation; however, they can often be grouped into the following general categories:

1. Point Degradations
2. Spatial Degradations
3. Temporal Degradations
4. Chromatic Degradations
5. Combinations of the above [3]

While coordinate changes causes point degradations, temporal and chromatic degradations imply time and color deterioration of their respective axes. Spatial image degradations, which are the main concerns of this thesis, result in blurring of the image [2].

The main reasons for spatial degradations are as follows:

1. Diffraction effects in optical systems
2. first-, second-, and higher-order optical system aberrations
3. atmospheric turbulence
4. motion blur
5. defocused systems [3]

3.1. Nonlinear Image Restoration [2]

In this thesis the main concern is the restoration of the images blurred by nonlinear space-variant imaging systems. A discrete observation model will be derived representing the physical imaging system that has a blurring effect given in Figure 3-1.

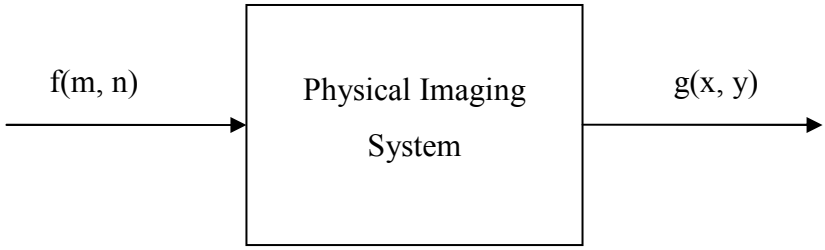

Figure 3-1. Physical Imaging System with blurring effect

We know from chapter 2 that the image formation is a neighborhood process, so that a point on the image plane depends on a region of neighboring points on the object plane. In other words the point (x, y) on the image plane is a function of the point (m, n) and its neighbor points of the object plane. Also the image point (x, y) is related with the object point (m, n) and its neighbor points by the point-spread function, h of the image formation system. As the distance between the neighbor points and the point (m, n) increases, the effects of the neighboring points on forming the image point (x, y) on the image plane decreases.

We should derive a discrete model for the formation of the blurred image for processing it on the computer.

3.1.1. The Discrete Model for the Nonlinear Blurred Image

Let us take a nonlinear space-variant imaging system which is defined by its nonlinear point-spread function given in equation (2.5). The input of this

system is assumed to be the original object distribution called the ideal image f(m, n) and the output is the blurred image g(x, y) as shown in Figure 3-1.

The approximation for equation (2.5) in the discrete domain can be derived by dividing the integration interval into infinitesimal discrete intervals represented by Δ. Then the equation (2.5) becomes

$$g(x,y)=\sum_{n=-\infty}^{\infty}\sum_{m=-\infty}^{\infty}h(x,y,m\Delta,n\Delta,f(m\Delta,n\Delta))\Delta^2, \qquad (3.1)$$

where x, y, m, n are integer values.

The discrete model given in equation (3.1) will be used throughout the chapter from now on. Also, a point g(x, y) in the blurred image plane is created by the image plane points given in the Figure 3-2.

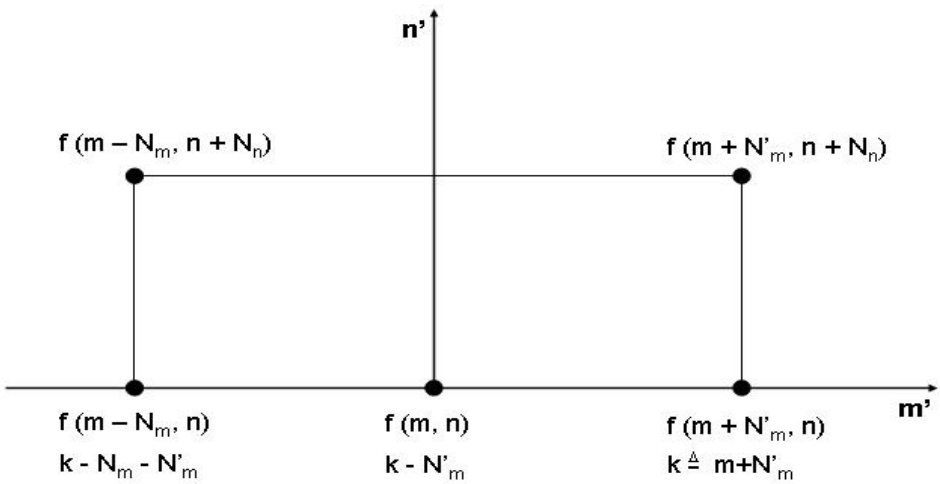

Figure 3-2. Neighborhood of the point (m, n) on the object plane

Therefore the image equation (3.1) can be written as the following:

$$
\begin{aligned}
g(x,y) &= h\left(x,y,\left(m+N_m^{'}\right)\Delta,n\Delta,f\left(\left(m+N_m^{'}\right)\Delta,n\Delta\right)\right)\Delta^2+\ldots \\
&+h\left(x,y,\left(m+N_m^{'}\right)\Delta,\left(n+N_n\right)\Delta,f\left(\left(m+N_m^{'}\right)\Delta,\left(n+N_n\right)\Delta\right)\right)\Delta^2 \\
&+h\left(x,y,\left(m+N_m^{'}-1\right)\Delta,n\Delta,f\left(\left(m+N_m^{'}-1\right)\Delta,n\Delta\right)\right)\Delta^2+\ldots \\
&+h\left(x,y,\left(m+N_m^{'}-1\right)\Delta,\left(n+N_n\right)\Delta,f\left(\left(m+N_m^{'}-1\right)\Delta,\left(n+N_n\right)\Delta\right)\right)\Delta^2+\ldots \\
&+h\left(x,y,\left(m-N_m\right)\Delta,n\Delta,f\left(\left(m-N_m\right)\Delta,n\Delta\right)\right)\Delta^2+\ldots \\
&+h\left(x,y,\left(m-N_m\right)\Delta,\left(n+N_n\right)\Delta,f\left(\left(m-N_m\right)\Delta,\left(n+N_n\right)\Delta\right)\right)\Delta^2 .
\end{aligned}
\tag{3.2}
$$

Equation (3.2) can be rewritten as

$$
g(x,y)=\sum_{i=-N_m}^{N_m^{'}}\sum_{j=0}^{N_m}h\left(x,y,(m+i)\Delta,(n+j)\Delta,f\left((m+i)\Delta,(n+j)\Delta\right)\right)\Delta^2
\tag{3.3}
$$

$$
=\sum_{i=m-N_m}^{m+N_m^{'}}\sum_{j=n}^{n+N_m}h\left(x,y,i\Delta,j\Delta,f\left(i\Delta,j\Delta\right)\right)\Delta^2
\tag{3.4}
$$

$$
=\sum_{i=0_m}^{N_m+N_m^{'}}\sum_{j=0}^{N_m}h\left(x,y,\left(m-N_m+i\right)\Delta,(n+j)\Delta,f\left(\left(m-N_m+i\right)\Delta,(n+j)\Delta\right)\right)\Delta^2 .
\tag{3.5}
$$

If we assume $k\underline{\underline{\Delta}}m+N_m^{'}$, equation (3.5) becomes

$$
g(x,y)=\sum_{i=k-N_m-N_m^{'}}^{k}\sum_{j=n}^{n+N_m}h\left(x,y,i\Delta,j\Delta,f\left(i\Delta,j\Delta\right)\right)\Delta^2 .
\tag{3.6}
$$

Rewriting the equation (3.8) yields

$$
\begin{aligned}
g(x,y) &= \sum_{j=n}^{n+N_n}h\left(x,y,k\Delta,j\Delta,f\left(k\Delta,j\Delta\right)\right)\Delta^2 \\
&+\sum_{j=n}^{n+N_n}h\left(x,y,(k-1)\Delta,j\Delta,f\left((k-1)\Delta,j\Delta\right)\right)\Delta^2 \\
&+\ldots \\
&+\sum_{j=n}^{n+N_n}h\left(x,y,\left(k-N_m-N_m^{'}\right)\Delta,j\Delta,f\left(\left(k-N_m-N_m^{'}\right)\Delta,j\Delta\right)\right)\Delta^2 .
\end{aligned}
\tag{3.7}
$$

Let us define the followings:

$$\sum_{j=n}^{n+N_n} h(x, y, k\Delta, j\Delta, f(k\Delta, j\Delta))\Delta^2 \underline{\underline{\Delta}} H_k(x, y, k\Delta, n\Delta, f(k\Delta, n\Delta))\Delta^2$$

$$\sum_{j=n}^{n+N_n} h(x, y, (k-1)\Delta, j\Delta, f((k-1)\Delta, j\Delta))\Delta^2$$

$$\underline{\underline{\Delta}} H_{k-1}(x, y, (k-1)\Delta, n\Delta, f((k-1)\Delta, n\Delta))\Delta^2$$

$$\ldots$$

$$\sum_{j=n}^{n+N_n} h(x, y, (k-N_m-N_m^{'})\Delta, j\Delta, f((k-N_m-N_m^{'})\Delta, j\Delta))\Delta^2$$

$$\underline{\underline{\Delta}} H_{k-N_m-N_m^{'}}(x, y, (k-N_m-N_m^{'})\Delta, n\Delta, f((k-N_m-N_m^{'})\Delta, n\Delta))\Delta^2 . \tag{3.8}$$

Let $$\underset{\sim}{f}(k,n) \underline{\underline{\Delta}} \begin{bmatrix} f(k, n+N_n) \\ f(k, n+N_n-1) \\ \ldots \\ f(k,n) \end{bmatrix}, \tag{3.9}$$

Therefore the observation equation becomes

$$g(x, y) = \sum_{i=k-N_m-N_m^{'}}^{k} h\left(x, y, i\Delta, n\Delta, \underset{\sim}{f}(i\Delta, n\Delta)\right)\Delta^2 . \tag{3.10}$$

3.1.2. Example

An observation model is derived by assuming that a point (x, y) on the blurred image g(x, y) is created by the points on the object plane as given in the Figure 3-3.

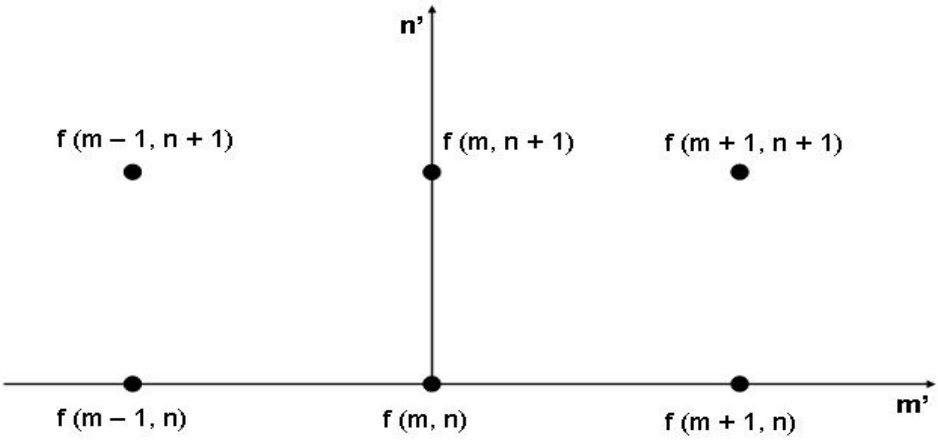

Figure 3-3. Ideal image points creating the blurred image point at (x, y)

Let us rewrite the equation (3.6) with the following parameter values:

$$N_m = 1 \ , \ N_m^{'} = 1, \ N_n = 1, \ \Delta = 1$$

$$g(x,y) = \sum_{i=k-2}^{k} \sum_{j=n}^{n+1} h(x,y,i,j,f(i,j)). \tag{3.11}$$

Also we know that $k\underline{\underline{\Delta}}m + N_m^{'} = m+1$, therefore, equation (3.11) becomes:

$$g(x,y) = \sum_{i=m-1}^{m+1} \sum_{j=n}^{n+1} h(x,y,i,j,f(i,j)) \tag{3.12}$$

$$g(x,y) = \sum_{i=-1}^{1} \sum_{j=0}^{1} h(x,y,i+m,j+n,f(i+m,j+n)). \tag{3.13}$$

If we rewrite the equation given above in explicit form, (3.13) becomes:

$$
\begin{aligned}
g(x,y) &= h(x,y,m+1,n,f(m+1,n)) \\
&+ h(x,y,m+1,n+1,f(m+1,n+1)) \\
&+ h(x,y,m,n,f(m,n)) \\
&+ h(x,y,m,n+1,f(m,n+1)) \\
&+ h(x,y,m-1,n,f(m-1,n)) \\
&+ h(x,y,m-1,n+1,f(m-1,n+1)).
\end{aligned} \tag{3.14}
$$

By using the definitions given in the equation (3.8), (3.14) becomes:

$$
\begin{aligned}
g(x,y) &= H_{m+1}\left(x,y,m+1,n,\underset{\sim}{f}(m+1,n)\right) \\
&+ H_{m}\left(x,y,m,n,\underset{\sim}{f}(m,n)\right) \\
&+ H_{m-1}\left(x,y,m-1,n,\underset{\sim}{f}(m-1,n)\right),
\end{aligned} \tag{3.15}
$$

$$
\text{where } \underset{\sim}{f}(k,n) \triangleq \begin{bmatrix} f(k,n+1) \\ f(k,n) \end{bmatrix}. \tag{3.16}
$$

The relation between the point (x, y+1) on the blurred image plane and the points on the object plane can be determined in the same manner.

$$
\begin{aligned}
g(x,y+1) &= H_{m+1}\left(x,y,m+1,n+1,\underset{\sim}{f}(m+1,n+1)\right) \\
&+ H_{m}\left(x,y,m,n+1,\underset{\sim}{f}(m,n+1)\right) \\
&+ H_{m-1}\left(x,y,m-1,n+1,\underset{\sim}{f}(m-1,n+1)\right),
\end{aligned} \tag{3.17}
$$

$$
\text{where } \underset{\sim}{f}(k,n) \triangleq \begin{bmatrix} f(k,n+2) \\ f(k,n+1) \end{bmatrix}. \tag{3.18}
$$

Putting these equations into matrix form yields the observation model for a blurred image point pair given below.

$$\begin{bmatrix} g(x,y+1) \\ g(x,y) \end{bmatrix} = \begin{bmatrix} H_{m+1}\left(x,y,m+1,n+1,\underset{\sim}{f}(m+1,n+1)\right) \\ H_{m+1}\left(x,y,m+1,n,\underset{\sim}{f}(m+1,n)\right) \end{bmatrix} +$$

$$\begin{bmatrix} H_{m}\left(x,y,m,n+1,\underset{\sim}{f}(m,n+1)\right) \\ H_{m}\left(x,y,m,n,\underset{\sim}{f}(m,n)\right) \end{bmatrix} +$$

$$\begin{bmatrix} H_{m-1}\left(x,y,m-1,n+1,\underset{\sim}{f}(m-1,n+1)\right) \\ H_{m-1}\left(x,y,m-1,n,\underset{\sim}{f}(m-1,n)\right) \end{bmatrix}. \tag{3.19}$$

Rewriting equation (3.19) in explicit form,

$$\begin{bmatrix} g(x,y+1) \\ g(x,y) \end{bmatrix} = \begin{bmatrix} h(x,y,m+1,n+1,f(m+1,n+1)) \\ h(x,y,m+1,n,f(m+1,n)) \end{bmatrix}$$

$$+\begin{bmatrix} h(x,y,m+1,n+2,f(m+1,n+2)) \\ h(x,y,m+1,n+1,f(m+1,n+1)) \end{bmatrix}$$

$$+\begin{bmatrix} h(x,y,m,n+1,f(m,n+1)) \\ h(x,y,m,n,f(m,n)) \end{bmatrix}$$

$$+\begin{bmatrix} h(x,y,m,n+2,f(m,n+2)) \\ h(x,y,m,n+1,f(m,n+1)) \end{bmatrix}$$

$$+\begin{bmatrix} h(x,y,m-1,n+1,f(m-1,n+1)) \\ h(x,y,m-1,n,f(m-1,n)) \end{bmatrix}$$

$$+\begin{bmatrix} h(x,y,m-1,n+2,f(m-1,n+2)) \\ h(x,y,m-1,n+1,f(m-1,n+1)) \end{bmatrix}. \tag{3.20}$$

CHAPTER 4

FIRST METHOD FOR STATE ESTIMATION

The state estimation algorithm called Optimum Decoding-Based Smoothing Algorithm [1], used for nonlinear image restoration is presented in this chapter.

4.1. Models and Assumptions [1]

The discrete models we deal with are given below:

State Transition Equation, $x(k+1) = f(k, x(k), w(k))$ (4.1)

Observation Equation, $z(k) = g(k, x(k), v(k))$

where x(0) is an n x 1 initial state random vector; x(k) is an n x 1 state vector at time k; w(k) is a p x 1 disturbance-noise vector at time k with zero mean and known statistical properties; v(k) is an l x 1 observation-noise vector at time k with zero mean and known statistics; z(k) is an r x 1 observation vector at time k. Time k is time $t_0 + kT_0$ where t_0 and T_0 are the initial time and the observation interval, respectively; $f(k, x(k), w(k))$ and $g(k, x(k), v(k))$ are linear or nonlinear vectors with appropriate dimensions. Furthermore, the random vectors x(0), w(j), w(k), v(l), v(m) are assumed to be independent for all j, k, l, m. The goal is to estimate the state sequence {x(0), x(1), …, x(L)} by using the observation sequence {z(1), z(2), …, z(L)}, where L is a chosen integer. This is a nonlinear estimation problem. The estimation algorithm presented in this chapter is applicable to any type of estimation problem, as long as the estimation problem under consideration can be modeled in Equation (4.1).

4.2. Quantization of States and Transition Probabilities [1]

A type of quantization for states and some difficulties in calculating transition probabilities between quantization levels are described in this section. If we consider the state x(k), it is a random vector whose range is in the space R^n (n-dimensional Euclidean space). Let us divide R^n into nonoverlapping subspaces R_i^n and assign a unique value x_{qi} to each subspace R_i^n, where the subscript q is for quantization.

Definition 1. A function $x_q(.) \underline{\underline{\Delta}} Q\{x(.)\}$ is a quantizer for the state x(.) if the followings hold:

1. $x_q(.) \underline{\underline{\Delta}} Q\{x(.)\} = x_{qi}$ whenever x(.) $\in R_i^n$; and
2. x_{qi} is unique for each R_i^n

Definition 2. The function $x_q(.)$ is the quantized state vector at time (.), and its possible values are sometimes called the quantization levels of the state x(.).

Definition 3. Subspace R_i^n is sometimes called gate (or cell) R_i^n.

Definition 4. The value x_{qi} is called the quantization level for the gate (cell) R_i^n.

Quantization means that whenever a random state vector x(.) falls within a given subspace, say R_i^n, the state x(.) is quantized to the unique value x_{qi}.

Definition 5. The transition probability $\Pi_{jm}(k)$ is the probability that the state x(k+1) will lie in the gate R_m^n when the state x(k) is in the gate R_j^n; i.e.,

$$\Pi_{jm}(k) \underline{\underline{\Delta}} \Pr ob\{x(k+1) \in R_m^n \mid x(k) \in R_j^n\}. \tag{4.2}$$

The quantization process and the transition probability can be seen schematically in Figure 4-1.

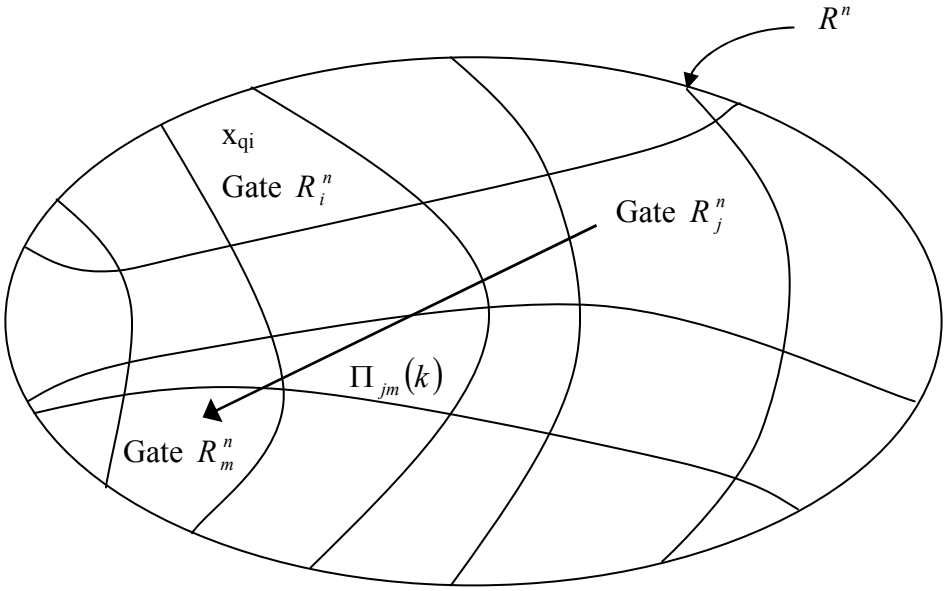

Figure 4-1. Quantization and Transition Probabilities

The transition probability $\Pi_{jm}(k)$ is a conditional probability. Hence it can be written as;

$$\Pi_{jm}(k) = \frac{\Pr ob\left\{x(k+1) \in R_m^n, x(k) \in R_j^n\right\}}{\Pr ob\left\{x(k) \in R_j^n\right\}} \tag{4.3}$$

$$\Pi_{jm}(k) = \left[\int_{R_j^n} p(x(k))dx(k)\right]^{-1} \times \left[\int_{R_j^n}\int_{R_m^n} p(x(k+1), x(k))dx(k+1)dx(k)\right]$$

$$\Pi_{jm}(k) = \left[\int_{R_j^n} p(x(k))dx(k)\right]^{-1} \times \left\{\int_{R_j^n}\left[\int_{R_m^n} p(x(k+1) \mid x(k))dx(k+1)\right]p(x(k))dx(k)\right\}$$

where p(x(k+1), x(k)) is the joint probability density function of x(k+1) and x(k); p(x(k)) is the probability density function of x(k); and p(x(k+1)|x(k)) is the conditional probability density function of x(k+1) given x(k). It is not usually easy to evaluate the transition probability $\Pi_{jm}(k)$ analytically. The difficulties are due to the shapes of the gates R_j^n and R_m^n and the statistics of the disturbance-noise vectors w(.) and the initial state vector x(0). The problem is more difficult if the state transition model is nonlinear. The next section discusses an approximate state transition model called the finite-state model which is obtained by approximating the disturbance noise vector w(k) and the initial state vector x(0) by discrete random vectors (see Appendix A), and by quantizing the state x(k) as previously described for all k = 1, 2, … . For this finite-state model, the transition probabilities can be easily calculated.

4.3. A Finite State Model [1]

The gates are assumed to be generalized rectangles such that the zero vector 0 (origin) is located in the center of a generalized rectangle, say R_0^n. Let the lengths of the sides of a generalized rectangle, say R_i^n, be g_{i1}, g_{i2}, …, g_{in}. These lengths are said to be the sizes of gate R_i^n. Moreover, the quantization levels for gates are assumed to be the center points of the gates, namely,

$$x_q(.) \underline{\underline{\Delta}} Q\{x(.)\} = x_{qi} \qquad \text{if } x(.) \in R_i^n, \tag{4.4}$$

where x_{qi} is the center of the generalized rectangle (gate) R_i^n.

Let us now define the finite-state model which approximates the state-transition model. For each k, the disturbance noise vector w(k) is approximated by a discrete random vector $w_d(k)$ whose possible values are $w_{d1}(k)$, $w_{d2}(k)$, …, $w_{dm_k}(k)$; the corresponding probabilities are $p_{d1}(k)$, $p_{d2}(k)$, …, $p_{dm_k}(k)$, i.e., $\Pr ob\{w_d(k) = w_{di}(k)\} = p_{di}(k)$, where i=1, 2, …,$m_k$ and m_k is a positive integer. Also, the initial state vector x(0) is approximated by a discrete random vector

$x_d(0)$ whose possible values are $x_{d1}(0)$, $x_{d2}(0)$, …, $x_{dn_0}(0)$; the corresponding probabilities are $p_{d1}(0)$, $p_{d2}(0)$, …, $p_{dn_0}(0)$, i.e., $\Pr ob\{x_d(0) = x_{di}(0)\} = p_{di}(0)$, where i=1, 2, …,$n_0$ and n_0 is a positive integer. The positive integers m_k and n_0 are chosen so that the random vectors w(k) and x(0) are satisfactorily approximated by the discrete vectors $w_d(k)$ and $x_d(0)$ for the considered estimation problem respectively. Further, in the state transition model, by replacing the disturbance noise vector w(k) and the initial state vector x(0) with the discrete random vectors $w_d(k)$ and $x_d(0)$, respectively, and then quantizing the states by equation (4.4), the state transition model is reduced to the finite-state model

$$x_q(k+1) = Q\{f(k, x_q(k), w_d(k))\}, \tag{4.5}$$

where $Q\{.\}$ is the quantizer; $x_q(k)$ is the quantized state vector at time k and its possible values, i.e., the quantization levels of the state vector $x(k)$ are $x_{q1}(k)$, $x_{q2}(k)$, …, $x_{qn_k}(k)$ where n_k is the number of possible quantization levels of the state vector $x(k)$; $x_q(0) \triangleq x_d(0)$ [by definition, $x_{qi}(0) \triangleq x_{di}(0)$, i=1, 2, …, n_0; in other words, the quantization levels of x(0) are assumed to equal the possible values of the discrete random vector x(0)]. Throughout this chapter whenever the state transition model (or the state transition equation) is mentioned we refer to equation (4.5); i.e., it is assumed that the state transition is described by the finite-state model.

The transition probability $\Pi_{jl}(k)$, which is defined by the conditional probability that the quantized state vector $x_q(k+1)$ will be equal to the quantization level x_{ql} for gate R_l^n, given that the quantized state vector $x_q(k)$ is equal to the quantization level x_{qj} for gate R_j^n, namely,

$$\Pi_{jl}(k) = \Pr ob\{x_q(k+1) = x_{ql} \mid x_q(k) = x_{qj}\} \tag{4.6}$$

is determined as follows:

Let us assume that the quantized state vector $x_q(k)$ is equal to the quantization level x_{qj} for gate R_j^n. The transitions from this quantization level to the others are determined by the discrete random vector $w_d(k)$ and the function $Q\{f(k, x_q(k) = x_{qj}, w_d(k))\}$. The discrete random vector $w_d(k)$ can take any value in the set $\{w_{d1}(k), w_{d2}(k), ..., w_{dm_k}(k)\}$ with corresponding probabilities $p_{d1}(k), p_{d2}(k), ..., p_{dm_k}(k)$. Thus, the quantized state vector $x_q(k+1)$ can be equal to at most m_k various quantization levels. If the function $f(k, x_q(k) = x_{qj}, w_d(k))$ maps x_{qj} into another gate, say R_i^n for only one possible value, say $w_{di}(k)$, of the discrete random vector $w_d(k)$, then the transition probability $\Pi_{ji}(k)$ from gate R_j^n to gate R_i^n is the probability that the possible value $w_{di}(k)$ of $w_d(k)$ occurs, i.e. , $\Pi_{ji}(k) = p_{di}(k)$. However, if the function $f(k, x_q(k) = x_{qj}, w_d(k))$ maps x_{qj} into another gate, say R_l^n, for more than one possible value, say $w_{d1}(k)$ and $w_{d2}(k)$ of $w_d(k)$, the transition probability $\Pi_{jl}(k)$ from gate R_j^n to gate R_l^n is the probability that the discrete random vector $w_d(k)$ i.e., equal to either of the possible values $w_{d1}(k)$ or $w_{d2}(k)$, i.e., $\Pi_{jl}(k) = \sum_n p_{dn}(k) = p_{d1}(k) + p_{d2}(k)$, where the summation is over all n such that $Q\{f(k, x_q(k) = x_{qj}, w_{dn}(k))\} = x_{ql}$. Having determined the finite-state model, we can represent the target motion by a trellis diagram.

4.4. Trellis Diagram for the State Model [1]

Let us assume that the quantized state vector $x_q(k)$ has n_k possible values, say $x_{q1}(k)$, $x_{q2}(k)$, …, $x_{qn_k}(k)$ where n_k is a positive integer. To represent the state transition model by a graph, we adopt the following conventions:

1. Each possible value of $x_q(k)$ is represented on the kth column by a point (sometimes called node) with the corresponding quantization level so that the kth column contains the possible quantization levels

of $x(k)$ (in other words, the possible gates in which the target can lie at time (k)) where k=0, 1, 2, ….

2. The transition from one quantization level to another is represented by a line having a direction indicating the direction of the state transition.

Hence, the state transition from time zero to time L can be represented by a directed graph shown in Figure 4-2, which is called the trellis diagram for the state transition from time zero to time L.

Definition 6. A path in the trellis diagram any sequence of directed lines where the final vertex of one is the initial vertex of the next.

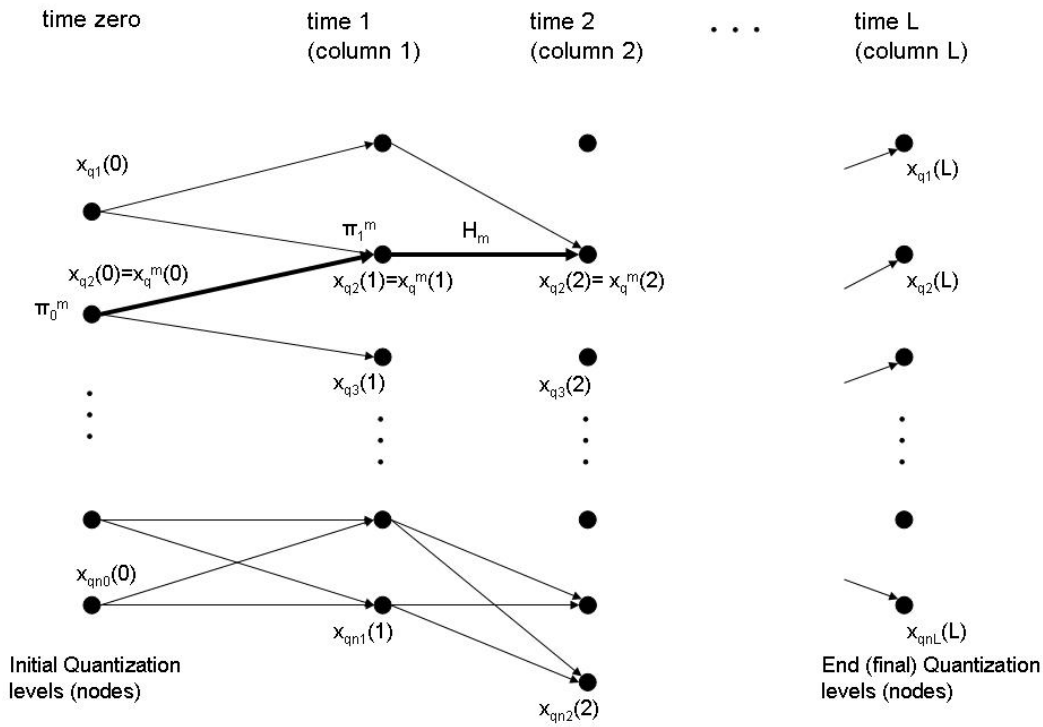

Figure 4-2. The trellis diagram for the state transition

4.5. Approximate Observation Models [1]

So far the state transition model has been reduced to a finite-state model which uses the quantized state vector $x_q(.)$. However, the observation model in equation (4.1) uses the state vector $x(.)$. Thus, in the observation model in equation (4.1), by replacing the state vector x(k) with the quantized state vector $x_q(k)$, the following approximate observation model is obtained:

$$z(k) = g(k, x_q(k), v(k)) \tag{4.7}$$

From now on, when we discuss the observation model (or the measurement model equation) we refer to the model in (4.7) which is used in the following analyses.

Let us consider the trellis diagram in Figure 4-2, the state estimation process will be performed from time zero up to and including time L. Therefore, the trellis diagram is drawn from time zero to time L. Time zero refers to the initial state. Let us define the following symbols which are used for further analyses:

n_i :Number of quantization levels for the gates in which the target may lie at time i; in other words, the number of possible values of the quantized state vector $x_q(i)$ where i = 0, 1, 2, …, L.

$\underset{\sim}{x}(i)$:Set of all the quantization levels for the gates in which the target may lie at time i, namely, $\underset{\sim}{x}(i) \underline{\underline{\Delta}} \{x_{q1}(i), x_{q2}(i), ..., x_{qn_i}(i)\}$, where i = 0, 1, 2, …,L.

M :Number of possible paths through the trellis diagram. This number is equal to or less than $\prod_{j=0}^{L} n_j$.

H_m :The mth path through the trellis diagram, indicated by a bold line in Figure 4-2.

$x_q^m(i)$:Quantization level for the gate in which the state vector lies at time k when it follows path H_m. In other words, the possible value of the

quantized state vector $x_q(i)$ through which the mth path passes. For example, in trellis diagram of Figure 4-2, $x_q^m(0) = x_{q2}(0)$; $x_q^m(1) = x_{q2}(1)$; $x_q^m(2) = x_{q2}(2)$; …

Π_0^m :Probability that the possible value of the initial state vector $x_d(0)$ from which the mth path starts occurs, namely, $\Pi_0^m = \Pr ob\{x_d(0) = x_q^m(0)\}$. For example, in trellis diagram of Figure 4-2, $\Pi_0^m = \Pr ob\{x_q(0) = x_{q2}(0)\}$.

Π_i^m :Transition probability from the (i-1)th gate for the mth path. In other words, it is the transition probability that the state vector will be at the ith quantization level (node) of path H_m at time i when it is at the (i-1)th quantization level (node) of path H_m at time i-1; that is $\Pi_i^m \triangleq \Pr ob\{x_q(i) = x_q^m(i) \mid x_q(i-1) = x_q^m(i-1)\}$.

$\Pi_0^{\max}$:Maximum of the probabilities that the quantization levels at time zero occur.

$\Pi_i^{\max}$:Maximum of the transition probabilities from the quantization levels at time i-1 to the quantization levels at time i (where i = 1, 2, …, L).

$\Pi_0^{\min}$:Minimum of the probabilities that the quantization levels at time zero occur.

$\Pi_i^{\min}$:Minimum of the transition probabilities from the quantization levels at time i-1 to the quantization levels at time i (where i = 1, 2, …, L).

$\underset{\sim L}{x^m} \triangleq \{x_q^m(0), x_q^m(1), ..., x_q^m(L)\}$:Sequence of the quantization levels (nodes) which the mth path passes through; obviously, $x_q^m(i) \in \underset{\sim}{x}(i)$, where i = 0, 1, 2, …, L

$z^L = \{z(1), z(2), ..., z(L)\}$:Observation sequence from time 1 to time L

Obviously, the real state transition occurs along one of the possible paths through the trellis diagram. Hence our aim is to decide upon a path through the

trellis diagram which is most likely (probably) followed by the real state variable by using the observation sequence z^L. Because of randomness in the models, our approach must be statistical, i.e., a statistical optimization problem. Based on the observations, we shall guess which path was (most likely) followed by the real state variable. Hence, a criterion is needed. A suitable criterion may be the minimum error probability criterion. Using this criterion reduces the problem of finding the path most likely followed by the real state variable to a multiple-(composite) hypothesis-testing problem.

4.6. Minimum Error Probability Criterion [1]

In the previous section, M possible paths through the trellis diagram H_1, H_2, ..., H_M were labeled. These paths are sometimes referred to as hypotheses. Hence, we would like to decide which hypothesis is true by using the minimum error probability criterion and the observation sequence. We develop a decision rule that assigns each point in the observation space D to one of the hypotheses. The decision rule divides the whole observation space D into M subspaces D_1, D_2, ..., D_M. If the observations fall in the subspace D_i, we decide that the true path is H_i. Subspace D_i is called the decision region for hypothesis H_i. The decision regions, therefore, must be chosen in such a way that the overall probability is minimized.

The overall error probability, sometimes called the Bayes risk R, is defined by

$$R \underline{\underline{\Delta}} \sum_{j=1}^{M} \sum_{\substack{i=1 \\ i \neq j}}^{M} \left\{ \int_{z^L \in D_i} p(H_j) p(z^L \mid H_j) dz^L \right\} \tag{4.8}$$

where

$p(H_j)$:Probability that the hypothesis H_j (path H_j) is true. This is called the a priori probability of hypothesis H_j.

$p(z^L \mid H_j)$:Conditional probability of the observation sequence z^L given that hypothesis H_j is true.

In order to find the optimal decision rule, the decision regions $D_1, D_2, \ldots, D_M$ are varied so that the risk R is minimized. The optimum decision rule is;

choose H_i if $p(H_i)p(z^L \mid H_i) > p(H_j)p(z^L \mid H_j)$ for all j≠i. (4.9)

For a given observation sequence z^L, if the inequality in equation (4.9) becomes an equality for one or more hypotheses H_j, anyone of these H_j or H_i can be chosen as the decision. This does not change the average error probability.

Convention: throughout this chapter for all observation sequences z^L for which the inequality in equation (4.9) becomes an equality, the decision is made at random (e.g., by the flip *of* a coin) among the hypotheses satisfying the equality.

Hence the optimum decision region for hypothesis H_i becomes

$D_i \triangleq \{z^L : p(H_i)p(z^L \mid H_i) > p(H_j)p(z^L \mid H_j)\}$ for all j≠i. (4.10)

and all the observation sequences z^L for which the inequality in equation (4.9) becomes an equality, and then the hypothesis H_i is chosen. Note that the decision regions are nonoverlapping, namely, $D_i \cap D_j = \emptyset$ for j≠I, where ∩ and Ø stand for intersection and empty set, respectively. However union of all the decision regions covers the whole observation space D. Hence the optimum decision rule may be interpreted as "if the observation sequence z^L falls within the optimum decision region D_i, then choose hypothesis H_i as the decision; i.e.,

choose H_i if $z^L \in D_i$.

4.7. Optimum Decision Rule for State Transition Paths [1]

Let us consider the state transition model in equation (4.5) and the observation model in equation (4.7). The a priori probability of hypothesis H_i can be written as:

$$p(H_i) = \prod_{k=0}^{L} \Pi_k^i \qquad (4.11)$$

since the disturbance noise vector w(k) is assumed to be independent of w(j) and x(0)for all j≠k, where Π_k^i is as defined in section 4.5. Moreover, recognizing that the sequence $\underset{\sim L}{x^i}$ defined in section 4.5 describes hypothesis H_i completely and using equation (4.11) and the assumption that the observation noise is independent from sample to sample, the function $p\left(z^L \mid H_j\right)$ in (4.9) can be rewritten as;

$$p\left(z^L \mid H_j\right) = p\left(z^L \mid \underset{\sim L}{x^i}\right) = \prod_{k=1}^{L} p\left(z(k) \mid x_q^i(k)\right), \tag{4.12}$$

where $p\left(z(k) \mid x_q^i(k)\right)$ is the conditional probability of the observation z(k) given that $x_q(k) = x_q^i(k)$, i.e., $p\left(z(k) \mid x_q^i(k)\right) \underline{\underline{\Delta}} p\left(z(k) \mid x_q(k) = x_q^i(k)\right)$

Substituting equations (4.11) and (4.12) into the optimum decision rule of equation (4.9), we obtain the following:

$$\text{Choose } H_i \text{ if} \quad \Pi_0^i \prod_{k=1}^{L} \Pi_k^i p\left(z(k) \mid x_q^i(k)\right) > \Pi_0^j \prod_{k=1}^{L} \Pi_k^j p\left(z(k) \mid x_q^j(k)\right)$$

$$\text{for all } j \neq i. \tag{4.13}$$

Since it is frequently more convenient to perform summations than multiplications, and the natural logarithm function is monotonically increasing, taking the natural logarithm of both sides of the inequalities in equation (4.13) we get the following:

$$\text{Choose } H_i \text{ if} \quad \ln\left(\Pi_0^i\right) + \sum_{k=1}^{L} \left\{\ln\left(\Pi_k^i\right) + \ln\left(p\left(z(k) \mid x_q^i(k)\right)\right)\right\} >$$

$$\ln\left(\Pi_0^j\right) + \sum_{k=1}^{L} \left\{\ln\left(\Pi_k^j\right) + \ln\left(p\left(z(k) \mid x_q^j(k)\right)\right)\right\} \text{ for all } j \neq i. \tag{4.14}$$

Either one of the above equations (4.13) and (4.14) with the convention given in section 4.6 is the optimum decision rule for deciding the path most probably followed by the real state variable.

Let us now present some definitions to be used later in the chapter:

Definition 7. An initial node is a quantization level at time zero. The metric denoted by $MN(x_{qi}(0))$, of the initial node $x_{qi}(0)$ is defined by

$$MN(x_{qi}(0)) = \ln[\Pr ob\{x_q(0) = x_{qi}(0)\}] \tag{4.15}$$

consequently, $MN(x_q^m(0)) = \ln \Pi_0^m$

Definition 8. The metric, denoted by $M(x_{qj}(k-1) \to x_{qi}(k))$, of the branch which connects the quantization level (node) $x_{qj}(k-1)$ to the quantization level $x_{qi}(k)$ is defined by:

$$M(x_{qj}(k-1) \to x_{qi}(k)) \triangleq \ln[\Pr ob\{x_q(k) = x_{qi}(k) \mid x_q(k-1) = x_{qj}(k-1)\}]$$
$$\ln[p(z(k) \mid x_{qi}(k))] \tag{4.16}$$

Definition 9. The metric of a path from time zero to time i is the summation of the metric of the initial node from which the path starts and the metrics of the branches of which the path consists. For example, the metric, denoted by $M(x_q^m(i))$, of the portion between the nodes $x_q^m(0)$ and $x_q^m(i)$ of the path (hypothesis) H_m is

$$M(x_q^m(i)) = \ln \Pi_0^m + \sum_{k=1}^{i} [\ln \Pi_k^m + \ln p(z(k) \mid x_q^m(k))] \tag{4.17}$$

consequently, the metric, sometimes denoted by $M(H_m)$, of the path H_m (through the trellis) is

$$M(H_m) \triangleq M(x_q^m(L)) = \ln[p(H_m) p(z^L \mid H_m)] \tag{4.18}$$

where $x_q^m(L)$ is the end node of the path H_m, and $p(H_m)$ and $p(z^L \mid H_m)$ are given by equations (4.11) and (4.12).

Definition 10. The density function of the of the observation sequence z^L when the state variable actually followed the path H_m, i.e., $p(z^L \mid H_m)$ is referred to as the likelihood function for the path (hypothesis) H_m.

It follows from the definition 9 and the equation (4.14) that the optimum decision rule is to choose the path with the largest metric through the trellis diagram as the decision.

4.8. Optimum Decoding Based Smoothing Algorithm [1]

Preliminary step. Reduce the state transition model to a finite-state model, as described before, and obtain a trellis diagram for the state transition (model) from time zero to the time, say L, until which the state transitions will be tracked. Then assign to each initial node its metric.

Step 1. For each node at time 1; use the observation z(l) and evaluate the metrics of the branches connecting the initial nodes to the node at time 1. Add these metrics to the metrics of the initial nodes from which the branches start, find the metrics of the paths merging at the node at time 1, label the path with the largest metric (which is called the best path for the node at time 1), and then discard the other paths. Finally, assign the largest metric to the node at time 1 (which is called the metric of the node at time 1).

Step k. For each node at time k; use the observation at time k, [i.e., z(k)], and calculate the metrics of the branches connecting the nodes at time (k–1) to the node at time k. Add these metrics to the metrics of the nodes at time (k-1) from which the branches start, find the metrics of the paths merging at the node at time k, label the path with the largest metric (which is called the best path for the node at time k), and then discard the other paths. Finally, assign the largest metric to the node at time k (which is called the metric of the node at time k).

At the end of step L, stop and choose from among the nodes at time L that with the largest metric. Then decide that the best path for this node is the path followed by the state transitions.

4.8.1. An Example of the ODSA

Let us consider a state vector whose motion from time zero to time 2 is described by figure 4-3 using the ODSA, we would like to find the path in the trellis diagram which was most likely followed by the state transitions from time zero to time 2.

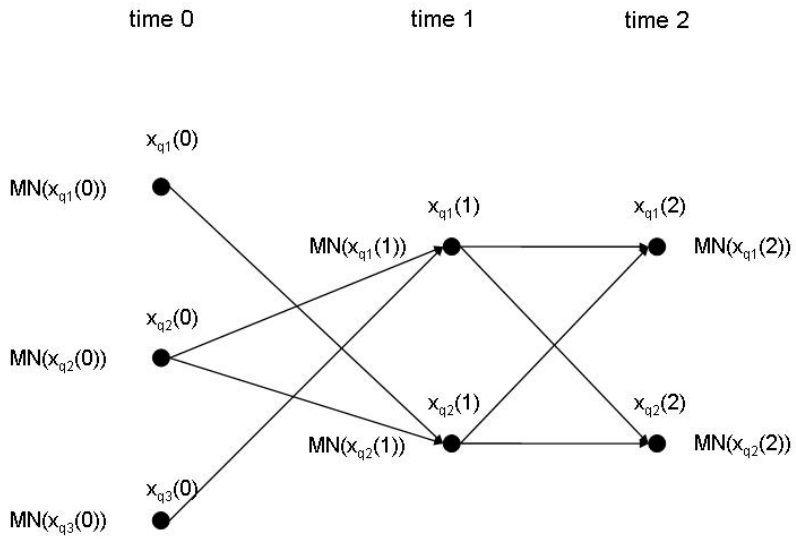

Figure 4-3. The trellis diagram for the ODSA example

Preliminary step. To each intial node, assign its metric, i.e., $MN(x_{qi}(0)) = \ln[\Pr ob(x_q(0) = x_{qi}(0))]$, where i=1, 2, 3. From now on, the metric of the node $x_{qi}(k)$ is represented by $MN(x_{qi}(k))$.

Step 1. Consider the node $x_{q1}(1)$. The branches $x_{q2}(0)x_{q1}(1)$ and $x_{q3}(0)x_{q1}(1)$ are the only ones connecting the nodes at time zero to the node $x_{q1}(1)$. Hence calculate the metrics of these branches, then add these metrics to the metrics of the nodes $x_{q2}(0)$ and $x_{q3}(0)$ and obtain the following:

$$A_{11} \underline{\underline{\Delta}} M\left(x_{q2}(0) \to x_{q1}(1)\right) + MN\left(x_{q2}(0)\right),$$
$$A_{12} \underline{\underline{\Delta}} M\left(x_{q3}(0) \to x_{q1}(1)\right) + MN\left(x_{q3}(0)\right). \tag{4.19}$$

Further, assuming that $A_{11} \geq A_{12}$ the path $x_{q2}(0)x_{q1}(1)$ is chosen as the best path for the node $x_{q1}(1)$, and A_{11} is assigned to the node $x_{q1}(1)$ as its metric, i.e., $MN\left(x_{q1}(1)\right) = A_{11}$. The path $x_{q3}(0)x_{q1}(1)$ is then discarded. Let us now assume that the followings are similarly found for the node $x_{q2}(1): x_{q1}(0)x_{q2}(1)$ is the best path for $x_{q2}(1)$, and $MN\left(x_{q2}(1)\right) = M\left(x_{q1}(0) \to x_{q2}(1)\right) + MN\left(x_{q1}(0)\right)$.

Step 2. Consider the node $x_{q1}(2)$. The branches $x_{q1}(1)x_{q1}(2)$ and $x_{q2}(1)x_{q1}(2)$ are those connecting the nodes at time 1 to the node $x_{q1}(2)$. Hence calculating the metrics of these branches and adding these metrics to the metrics of the nodes $x_{q1}(1)$ and $x_{q2}(1)$, we obtain the following:

$$A_{21} \underline{\underline{\Delta}} M\left(x_{q1}(1) \to x_{q1}(2)\right) + MN\left(x_{q1}(1)\right),$$
$$A_{22} \underline{\underline{\Delta}} M\left(x_{q2}(1) \to x_{q1}(2)\right) + MN\left(x_{q2}(1)\right). \tag{4.20}$$

Further, assuming that $A_{22} \geq A_{21}$ the path $x_{q1}(0)x_{q2}(1)x_{q1}(2)$ is chosen as the best path for the node $x_{q1}(2)$, and A_{22} is assigned to the node $x_{q1}(2)$ as its metric, i.e., $MN\left(x_{q1}(2)\right) = A_{22}$. The path $x_{q2}(0)x_{q1}(1)x_{q1}(2)$ is the discarded. Let us now assume that the followings are similarly found for the node $x_{q2}(2): x_{q2}(0)x_{q1}(1)x_{q2}(2)$ is the best path for $x_{q2}(2)$, and $MN\left(x_{q2}(2)\right) = M\left(x_{q1}(1) \to x_{q2}(2)\right) + MN\left(x_{q1}(1)\right)$. In addition, assuming that $MN\left(x_{q2}(2)\right) \geq MN\left(x_{q1}(2)\right)$, the path $x_{q2}(0)x_{q1}(1)x_{q2}(2)$ is chosen as the best

path followed by the state vector from time zero to time 2. And the quantized values in the chosen path are our estimates for the state vector.

4.8.2. The Metric of a Branch

Let us give the state transition and observation models in the presence of Gaussian disturbance and observation noises:

$$x(k+1) = f(k, x(k), w(k)), \quad \text{State Transition Model} \qquad (4.21)$$

$$z(k) = g(k, x(k)) + v(k), \quad \text{Observation Model} \qquad (4.22)$$

where x(0) is an n x 1 initial state Gaussian random vector with mean m_0 and covariance R_0; w(k) is a p x 1 Gaussian disturbance noise vector with zero mean and covariance $R_w(k)$; x(k) is an n x 1 state vector at time k; z(k) is an r x 1 observation vector at time k; f(k, x(k),w(k)) and g(k, x(k)) are linear or nonlinear vectors with appropriate dimensions; and v(k) is an r x 1 Gaussian observation noise vector with zero mean and covariance $R_v(k)$. Moreover, the random vectors x(0), w(j), w(k), v(l) and v(m) are assumed to be independent for all j, k, l, m.

The z(k) in the observation model given in (4.22) is a linear function of the Gaussian observation noise v(k). Hence given that $x(k) = x_q^i(k)$, the conditional probability density function of z(k) is a multivariate Gaussian density function. Thus we have

$$p\left(z(k) \middle| x_q^i(k)\right) \triangleq p\left(z(k) \middle| x(k) = x_q^i(k)\right)$$

$$= \frac{\exp - \left\{ \dfrac{\left[z(k) - g\left(k, x_q^i(k)\right)\right]^T R_v^{-1}(k) \left[z(k) - g\left(k, x_q^i(k)\right)\right]}{2} \right\}}{(2\Pi)^{r/2} \left[\det R_v(k)\right]^{1/2}} \qquad (4.23)$$

Substituting this into equation (4.16) yields the metric of the branch between the nodes $x_q^i(k-1)$ and $x_q^i(k)$, that is

$$M\left(x_q^i(k-1) \rightarrow x_q^i(k)\right) = \ln \Pi_k^i - \ln\left\{(2\Pi)^{r/2}\left[\det R_v(k)\right]^{1/2}\right\} - \frac{\left[z(k) - g\left(k, x_q^i(k)\right)\right]^T R_v^{-1}(k)\left[z(k) - g\left(k, x_q^i(k)\right)\right]}{2} \quad (4.24)$$

4.9. The Modified Version of the ODSA

For finding the state transitions from time zero to time L by using the Optimum Decoding Based Smoothing Algorithm, deciding the most probable path is accomplished by finding the path with the largest metric through a trellis diagram from time zero to time L. The ODSA does this by using the Viterbi decoding algorithm, which systematically examines all possible paths in the trellis diagram. Hence, if number of possible paths in the trellis diagram is very large, the ODSA requires a huge amount of memory and computation. [1]

To reduce the needed memory and computation time for the ODSA [1], the number of nodes that can be stored at any time instant k is limited, that is, during transitions the size of the array holding all possible $x_q(k)$ values is limited with an integer value. Let us say at a given time instant k, there are N_m possible nodes on the trellis diagram, i.e., $x_{qi}(k)$ and i=1, 2, …, N_m. But we have a limit, let us say N_n, on the number of nodes that can be stored in the memory at a given time instant k. If $N_n < N_m$, the nodes are put in order according to their metric values. Then N_n of the nodes that have the largest metric values are chosen and stored in the memory. The next transition is performed by using only the selected nodes. The rest, N_m - N_n, of the nodes are discarded and therefore, the number of possible paths is reduced.

4.10. The complexity analysis of the Modified Version of the ODSA

Let the maximum computation time for all the calculations for a single node at any given time instant k, be equal to t_n seconds. Also, let the number of possible states of the process noise w(k) be N_w at any time instant k. Since the number of nodes at a given time instant k is limited with an integer number, let us say N_n, assuming this limit has been already reached, at the next time instant k+1, the maximum number of nodes may become $N_n \times N_w$. After computing $N_n \times N_w$ nodes they are put in an order according to their metric values. Then N_n of them having the largest metric values are kept and the rest are discarded. Therefore, the computation of the following nodes will again produce $N_n \times N_w$ nodes. Hence, for a single step from time instant k to k+1, the upper bound for the calculation time becomes $t_n \times N_n \times N_w$ seconds. Additionally, the upper bound for the calculation time from time instant 0 to time instant L is $L \times t_n \times N_n \times N_w$. As a result, the upper bound for the total computation time for the modified version of ODSA is directly proportional to the observation interval, L; the limit on the number of nodes, N_n and the number of possible states of the process noise w(k), N_w.

CHAPTER 5

SECOND METHOD FOR STATE ESTIMATION

The second method used for nonlinear image restoration in this thesis is called Bootstrap Filter Algorithm [18]. This algorithm is a way of representing and recursively generating an approximation to the state probability density function (PDF) [19]. The main idea is to represent the required PDF as a set of random samples over state space [20-22]. As the number of samples becomes very large, they effectively provide an exact, equivalent, representation of the required PDF. Estimates of moments (such as mean and covariance) of percentiles of the state vector PDF can be obtained directly from the samples [19].

5.1. Problem Statement [19]

For the discrete time estimation problem, the state vector $x_k \in R^n$ is assumed to evolve according to the following system model

$$x_{k+1} = f_k(x_k, w_k) \tag{5.1}$$

where $f_k : R^n \times R^m \to R^n$ is the system transition or state transition function and $w_k \in R^n$ is a zero mean, white-noise sequence independent of past and current states. The PDF of w_k is assumed to be known. At discrete time, measurements $y_k \in R^p$ become available. These measurements are related to the state vector via the observation equation

$$y_k = h_k(x_k, v_k) \tag{5.2}$$

where $h_k : R^n \times R^r \to R^p$ is the measurement or observation function and $v_k \in R^r$ is another zero mean, white-noise sequence of known PDF, independent of past and present states and the system noise. It is assumed that the initial PDF

$p(x_1 \mid D_0) \equiv p(x_1)$ of the state vector is available together with the functional forms f_i and h_i for i=1,…,k. The available information at time step k is the set of measurements $D_k = \{y_i : i = 1,...,k\}$.

The requirement is to construct the PDF of the current state x_k, given all the available information: $p(x_k \mid D_k)$. In principle this PDF may be obtained recursively in two stages: prediction and update. Suppose that the required PDF $p(x_{k-1} \mid D_{k-1})$ at time step k-1 is available. Then using the system model it is possible to obtain the prior PDF of the state at time step k

$$p(x_k \mid D_{k-1}) = \int p(x_k \mid x_{k-1}) p(x_{k-1} \mid D_{k-1}) dx_{k-1} \tag{5.3}$$

Here the probabilistic model of the state evolution, $p(x_k \mid x_{k-1})$, which is a Markov model, is defined by the system equation and the known statistics of w_{k-1}

$$p(x_k \mid x_{k-1}) = \int p(x_k \mid x_{k-1}, w_{k-1}) p(w_{k-1} \mid x_{k-1}) dw_{k-1}$$

Since by assumption $p(w_{k-1} \mid x_{k-1}) = p(w_{k-1})$ we have

$$p(x_k \mid x_{k-1}) = \int \delta(x_k - f_{k-1}(x_{k-1}, w_{k-1})) \times p(w_{k-1}) dw_{k-1} \tag{5.4}$$

where $\delta(.)$ is the Dirac delta function. The delta function arises because if x_{k-1} and w_{k-1} are known, then x_k is obtained from a purely deterministic relationship (eqn. (5.1)). Then at time step k a measurement y_k becomes available and may be used to update the prior via Bayes rule

$$p(x_k \mid D_k) = \frac{p(y_k \mid x_k) p(x_k \mid D_{k-1})}{p(y_k \mid D_{k-1})} \tag{5.5}$$

where the normalizing denominator is given by

$$p(y_k \mid D_{k-1}) = \int p(y_k \mid x_k) p(x_k \mid D_{k-1}) dx_k \tag{5.6}$$

The conditional PDF of y_k given x_k, $p(y_k \mid x_k)$, is defined by the measurement model and the known statistics of v_k

$$p(y_k \mid x_k) = \int \delta(y_k - h_k(x_k, v_k)) p(v_k) dv_k \tag{5.7}$$

In the update equation, eqn. (5.5), the measurement y_k is used to modify the predicted prior from the previous time step to obtain the required posterior of the state.

The recurrence relations of eqns. (5.3) and (5.5) constitute the formal solution to the Bayesian recursive estimation problem. Analytic solutions to this problem are only available for a relatively small and restrictive choice of system and measurement models. Hence the need to find a method of implementation which allows the models to be constructed realistically rather than conveniently.

5.2. Bootstrap Filter Algorithm [19]

Suppose we have a set of random samples $\{x_{k-1}(i) : i = 1,..., N\}$ from the PDF $p(x_{k-1} \mid D_{k-1})$. The bootstrap filter is an algorithm for propagating and updating these samples to obtain a set of values $\{x_{k-1}(i) : i = 1,..., N\}$, which are approximately distributed as $p(x_k \mid D_k)$. Thus the filter is an approximate mechanism (simulation) of the relations in eqns. (5.3) and (5.5).

5.2.1. Prediction

Each sample is passed through the system model to obtain samples from the prior at time step k: $x_k^*(i) = f_{k-1}(x_{k-1}(i), w_{k-1}(i))$, where $w_{k-1}(i)$ is a sample drawn from the PDF of the system noise $p(w_{k-1})$.

5.2.2. Update

On receipt of the measurement y_k, evaluate the likelihood of each prior sample and obtain a normalized weight for each sample and obtain a normalized weight for each sample

$$q_i = \frac{p\left(y_k \mid x_k^*(i)\right)}{\sum_{j=1}^{N} p\left(y_k \mid x_k^*(j)\right)} \tag{5.8}$$

Thus define a discrete distribution over $\left\{x_k^*(i) : i = 1,\ldots,N\right\}$, with probability mass q_i associated with element i. Now resample N times from the discrete distribution to generate samples $\left\{x_k(i) : i = 1,\ldots,N\right\}$, so that for any j, $\Pr\left\{x_k(j) = x_k^*(i)\right\} = q_i$.

The above steps of prediction and update form a single iteration of the recursive algorithm. To initiate the algorithm, N samples $x_1^*(i)$ are drawn from the known prior $p(x_1)$. These samples feed directly into the update stage of the filter. We contend that the samples $x_k(i)$ are approximately distributed as the required PDF, $p(x_k \mid D_k)$.

The Bootstrap Filter Algorithm [18] is summarized below in Table 5.1.

Table 5-1. Bootstrap Filter Algorithm summary [18].

Initialization, (t=0) For i = 1,…, N, sample $x_0(i) \sim p(x_0)$ and set t = 1.
Importance Sampling Step For i = 1,…, N sample $\tilde{x}_t(i) \sim p(x_t \mid x_{t-1}(i))$ and set $\tilde{x}_{0:t}(i) = \left(x_{0:t-1}(i), \tilde{x}_t(i) \right)$. For i = 1,…, N, evaluate the importance weights $\tilde{w}_t(i) = p\left(y_t \mid \tilde{x}_t(i) \right)$. Normalize the importance weights.
Selection Step Resample with replacement N particles $(x_{0:t}(i); i = 1,..., N)$ from the set $\left(\tilde{x}_{0:t}(i); i = 1,..., N \right)$ according to the importance weights. Set $t \leftarrow t+1$ and go to step 2.

5.3. The Complexity Analysis of the Bootstrap Filter Algorithm

Let the computation time for a single sample be t_s. Therefore, from time instant 0 to time instant L the computation time for the whole estimation process will be $t_s \times L \times N_s$ where Ns is the total number of samples taken from the required PDF at any time instant k. But, the time required for the search algorithm for resampling is also effected by Ns. The timing performance for different values of Ns can be seen in the timing performance graphics in the following chapter.

CHAPTER 6

SIMULATIONS FOR NONLINEAR IMAGE RESTORATION

In this chapter, simulations of image restoration are presented using the method given in chapter 3. For each of the simulations, an image called ideal or original image, has been created according to the given state transition equations. The state values were used as pixel intensity values. Therefore, the state transition equations became the image model. After creating the original image, it has been degraded by using the nonlinear observation model. Also there exist zero mean additive Gaussian noises as disturbance noise in the state transition equations and as observation noise in the observation equation. After the blurring operation, the estimations of the original image have been found by using the modified version of the Optimum Decoding Based Smoothing Algorithm [1] and the Bootstrap Filter Algorithm [18].

For the generation of the ideal image, strips of pixels each having two pixel lines were used. Each strip was created by using two state transition equations, one for each pixel lines as shown in equation (5.1). The whole image, then, was formed by concatenating the independently created pixel strips vertically [2].

$$\begin{bmatrix} f(x+1, y+1) \\ f(x+1, y) \end{bmatrix} = \begin{bmatrix} G_1(f(x, y), f(x, y+1), w(x, y+1)) \\ G_2(f(x, y), f(x, y+1), w(x, y)) \end{bmatrix} \text{Image Model (6.1)}$$

In the image model G_1 and G_2 are arbitrary linear or nonlinear functions which relate the image points at locations (x, y) and (x, y+1) to the image points at locations (x+1, y) and (x+1, y+1). For the simulations all the image strips were

created using the same G_1 and G_2 function pair. Additionally, w(x, y) is the Gaussian disturbance noise with zero mean and known statistical properties at location (x, y) [2].

The blurred image was obtained by using the observation model after creating the ideal image. Then the observed image, i.e. blurred image, was used during the estimation of the ideal image. The algorithms used for image restoration can handle the estimation of causal systems [2]. Therefore, if we substitute $N_x^{'} = 0$, $N_x = 2$, $N_y = 1$ in equation (3.11) then the imaging system described by (3.19) becomes causal. As a result the observation model given in (6.2) is obtained [2].

$$\begin{aligned}
\begin{bmatrix} g(x,y+1) \\ g(x,y) \end{bmatrix} &= \begin{bmatrix} h(x,y,m,n+1,f(m,n+1)) \\ h(x,y,m,n,f(m,n)) \end{bmatrix} \\
&+ \begin{bmatrix} h(x,y,m,n+2,f(m,n+2)) \\ h(x,y,m,n+1,f(m,n+1)) \end{bmatrix} \\
&+ \begin{bmatrix} h(x,y,m-1,n+1,f(m-1,n+1)) \\ h(x,y,m-1,n,f(m-1,n)) \end{bmatrix} \\
&+ \begin{bmatrix} h(x,y,m-1,n+2,f(m-1,n+2)) \\ h(x,y,m-1,n+1,f(m-1,n+1)) \end{bmatrix} \\
&+ \begin{bmatrix} h(x,y,m-2,n+1,f(m-2,n+1)) \\ h(x,y,m-2,n,f(m-2,n)) \end{bmatrix} \\
&+ \begin{bmatrix} h(x,y,m-2,n+2,f(m-2,n+2)) \\ h(x,y,m-2,n+1,f(m-2,n+1)) \end{bmatrix} \quad \text{Observation Model (6.2)}
\end{aligned}$$

In the observation model, h(x, y, m, n, f(m, n)) is the nonlinear point spread function relating the intensity of the point (x,y) on the observed image with the intensity of the point (m, n) on the ideal image.

Assume that the following relations are defined [2]:

$$
\begin{aligned}
&x_1(k) \underline{\underline{\Delta}} f(x, y+1) \\
&x_2(k) \underline{\underline{\Delta}} f(x, y) \\
&x_1(k+1) \underline{\underline{\Delta}} f(x+1, y+1) \\
&x_2(k+1) \underline{\underline{\Delta}} f(x+1, y) \\
&w_1(k) \underline{\underline{\Delta}} w(x, y+1) \\
&w_2(k) \underline{\underline{\Delta}} w(x, y) \\
&h_1(k, x_1(k)) \underline{\underline{\Delta}} g(x, y+1) \\
&h_2(k, x_2(k)) \underline{\underline{\Delta}} g(x, y)
\end{aligned}
\tag{6.3}
$$

Then the image model becomes;

$$
\begin{bmatrix} x_1(k+1) \\ x_2(k+1) \end{bmatrix} = \begin{bmatrix} G_1(x_1(k), x_2(k), w_1(k)) \\ G_2(x_1(k), x_2(k), w_2(k)) \end{bmatrix} \qquad \text{Image Model[2]} \qquad (6.4)
$$

With the assumptions in (5.3), by adding the Gaussian process noises with zero mean and known statistics, $v_1(k)$ and $v_2(k)$, the observation model becomes;

$$
\begin{bmatrix} z_1(k) \\ z_2(k) \end{bmatrix} = \begin{bmatrix} h_1(k, x_1(k)) \\ h_2(k, x_2(k)) \end{bmatrix} + \begin{bmatrix} v_1(k) \\ v_2(k) \end{bmatrix} \qquad \text{Observation Model[2] (6.5)}
$$

where

$$
\begin{aligned}
&v_1(k) \underline{\underline{\Delta}} v(x, y+1) \\
&v_2(k) \underline{\underline{\Delta}} v(x, y)
\end{aligned}
. \tag{6.6}
$$

The simulations performed using the image and observation models given in the equations (6.4) and (6.5) are presented below.

To evaluate the error performance of the restoration process two values, called "mean_absolute_error" and "relative_error", were calculated as the following:

$$
mean\_absolute\_error = \frac{\sum_{image\_size} \left| x(k) - \hat{x}(k) \right|}{no\_of\_pixels\_of\_the\_original\_image}, \tag{6.7}
$$

where $\sum_{image\_size}\left|x(k)-\hat{x}(k)\right|$ is the sum of absolute error between the pixels of the original and the estimated images;

$$relative\_error = \frac{\sum_{image\_size}\left|x(k)-\hat{x}(k)\right|}{\sum_{image\_size}\left|x(k)\right|}\times 100 \tag{6.8}$$

where $\sum_{image\_size}\left|x(k)\right|$ is the sum of all pixel intensities of the original image.

For the simulations of nonlinear image restoration, original images of size 50 pixels by 100 pixels were created according to the state transition equations given in the following sections. The algorithms used for restoration of the blurred versions of the original images have some parameters that affect their estimation performance. The parameters used for the modified version of ODSA are denoted by the following variables:

max_x_k_size	:Maximum number of the state nodes stored at any sampling time, k
gate_size	:Gate size for the quantization of the values
no_of_sampling_points_x	:Number of possible values of the initial state variable (i.e. x(0))
no_of_sampling_points_w	:Number of possible values of the disturbance noise variable (i.e. w(k))

The parameter that affects the estimation performance of the Bootstrap Filter Algorithm is called Ns. Ns represents the number of samples taken from the probability density functions of the initial state, x(0) for initialization and the process noise, w(k) at any time instant k.

The models used for image creation and blurring of the original images are given in the following sections. Also the simulation results and performance analysis of each algorithm can be found in each section.

6.1. Model 1

The first image to be processed has the following state transition equations:

$$G_1(x_1(k), x_2(k), w_1(k)) = \left|7\cos^*(\Pi x_1(k))\right| + \left|10\sin^*(0.5\Pi x_2(k) + w_1(k))\right|, \tag{6.9}$$

*: cosine and sine of argument in radians

$$G_2(x_1(k), x_2(k), w_2(k)) = 3x_1(k) + \frac{x_2(k)}{|x_1(k)| + 1} + w_2(k). \tag{6.10}$$

and the point spread function of the pixels assumed to be:

$$h(x, y, m, n, f(m,n)) = \frac{(x+y)mnf(m,n)}{10}. \tag{6.11}$$

For this model the mean of the initial state, x(0) was assumed to be 20 and the variance of it was chosen as 1.75. The process noise, w(k) and the observation noise v(k) were taken as zero mean Gaussian random variables with variances 2.5 and 0.5 respectively. All the simulations for this model were performed by using MATLAB version 7 running on a computer with a Intel Pentium 4 – 2GHz processor and 256MB of RAM. For each simulation the estimation process for an image was performed for 250 times and the average of the results were calculated.

The original image and the blurred version of the original image according to the given models and statistics given above are presented in figures 6-1 and 6-2 respectively.

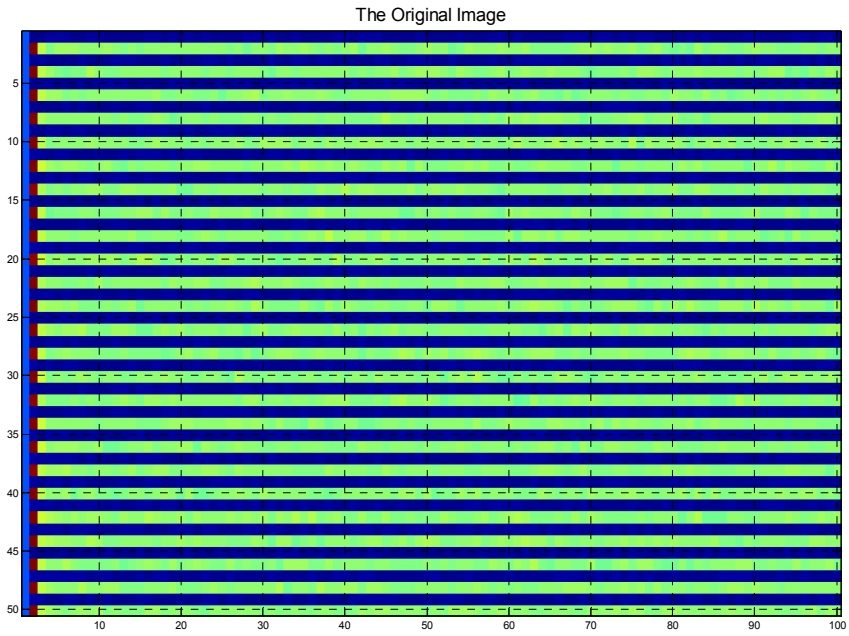

Figure 6-1 The Original Image according to the state transition equation

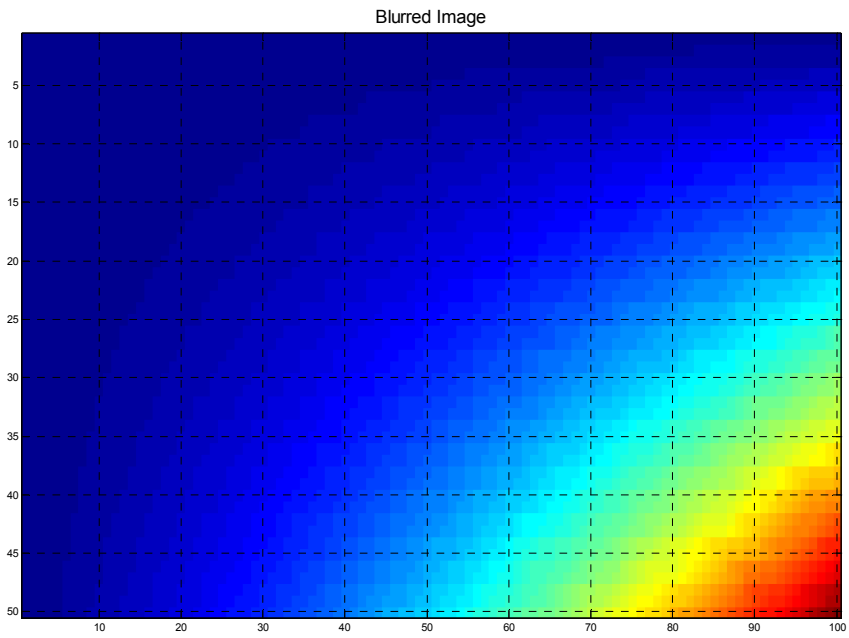

Figure 6-2 Result of the blurring function

6.1.1. Simulation 1

The results of the estimation algorithms with the parameters given below are as the following:

max_x_k_size	:9
gate_size	:0.1
no_of_sampling_points_x	:3
no_of_sampling_points_w	:3
Ns	:50

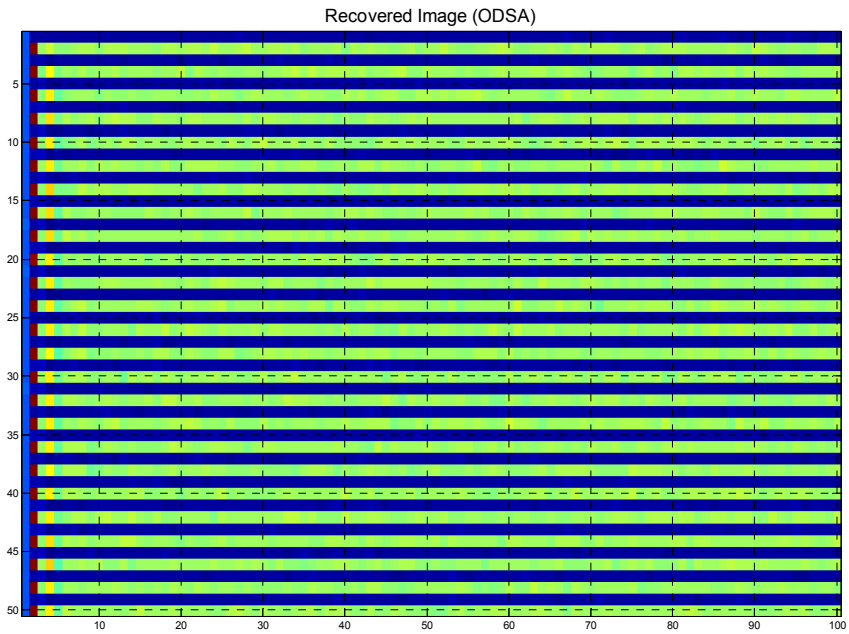

Figure 6-3 The Estimated Image by using the modified version of ODSA

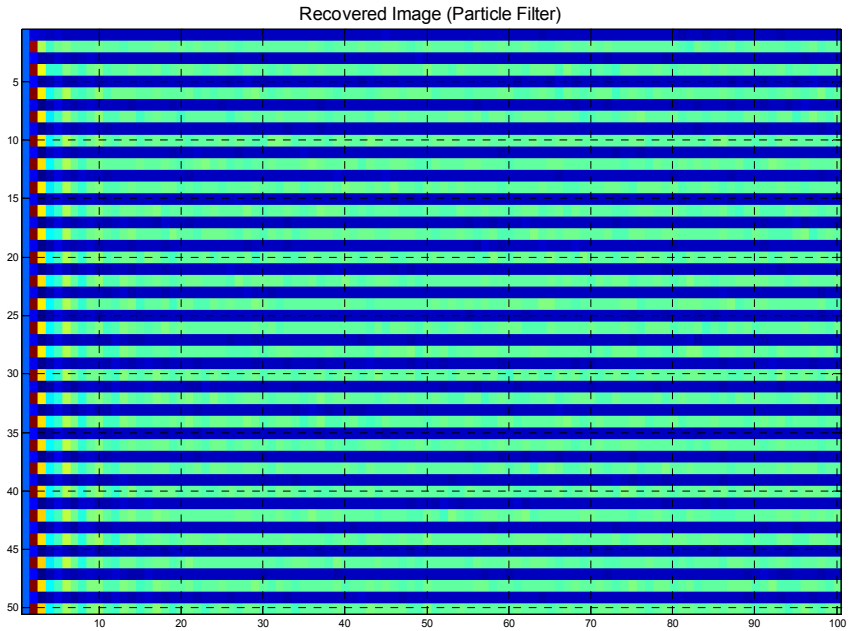

Figure 6-4 The Estimated Image by using the Bootstrap Filter Algorithm

The resulting mean_absolute_error for ODSA estimation is 6.5130 and the relative error is 27.65%. The time ODSA took for the estimation process is 8.5198 seconds per image.

When the Bootstrap Filter Algorithm was performed the mean_absolute_error became 6.4344 and the relative error was 27.32%. The processing time for the Bootstrap Filter was 7.5377 seconds per image.

6.1.2. Simulation 2

The second simulation for this model was performed with the following parameters:

max_x_k_size	:27
gate_size	:0.1
no_of_sampling_points_x	:3

no_of_sampling_points_w :3

Ns :100

The estimated images and the performance of the two algorithms are given below:

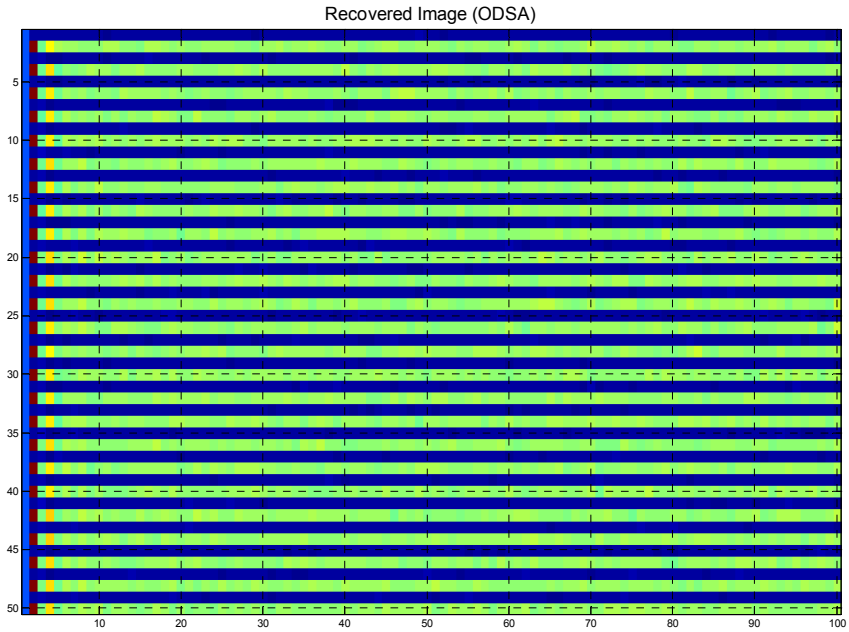

Figure 6-5 The Estimated Image by using the modified version of ODSA

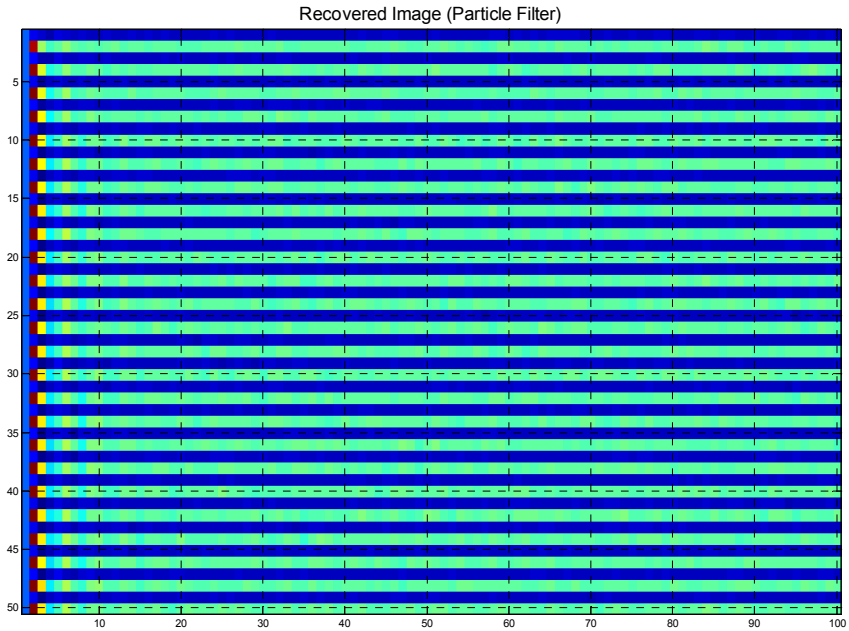

Figure 6-6 The Estimated Image by using the Bootstrap Filter Algorithm

The resulting mean_absolute_error for ODSA estimation is 6.3959 and the relative error is 27.15%. The time ODSA took for the estimation process is 25.4608 seconds per image.

When the Bootstrap Filter Algorithm was performed the mean_absolute_error became 6.3586 and the relative error was 27.01%. The processing time for the Bootstrap Filter was 23.3795 seconds per image.

6.1.3. Simulation 3

This time the parameters were set as the following:

max_x_k_size	:50
gate_size	:0.1
no_of_sampling_points_x	:3

no_of_sampling_points_w :3

Ns :150

With the parameters given above, the following results were obtained:

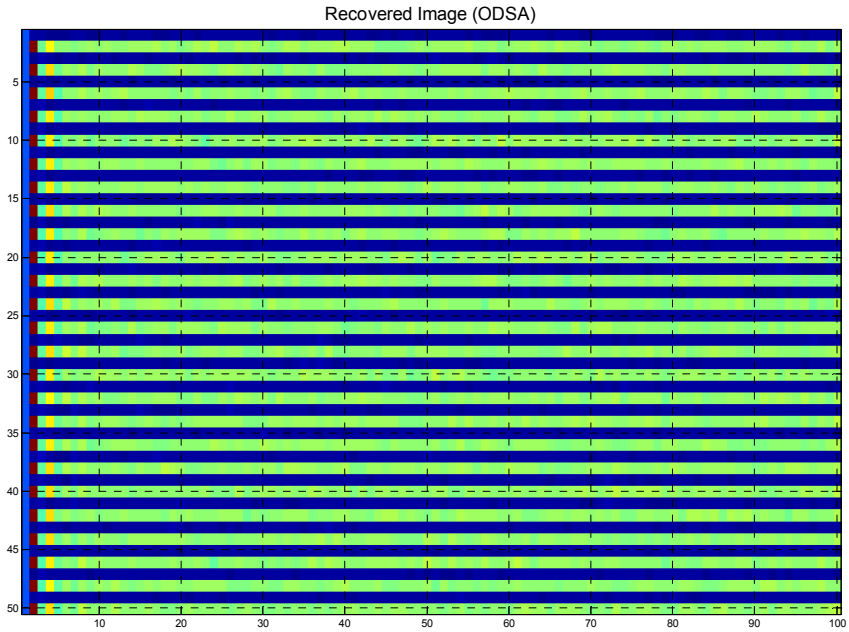

Figure 6-7 The Estimated Image by using the modified version of ODSA

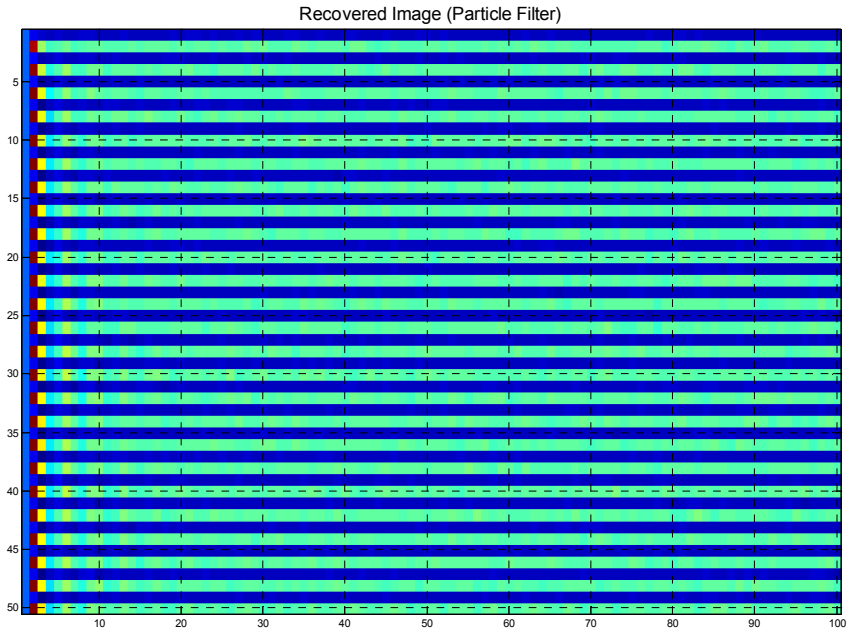

Figure 6-8 The Estimated Image by using the Bootstrap Filter Algorithm

The resulting mean_absolute_error for ODSA estimation is 6.4065 and the relative error is 27.18%. The time ODSA took for the estimation process is 60.6264 seconds per image.

With the Bootstrap Filter Algorithm the mean_absolute_error became 6.2701 and the relative error was 26.60%. The processing time for the Bootstrap Filter was 47.3703 seconds per image.

6.1.4. Simulation 4

For this simulation the parameters were changed to the following values:

max_x_k_size	:81
gate_size	:0.1
no_of_sampling_points_x	:3

no_of_sampling_points_w :3

Ns :250

The results are given below:

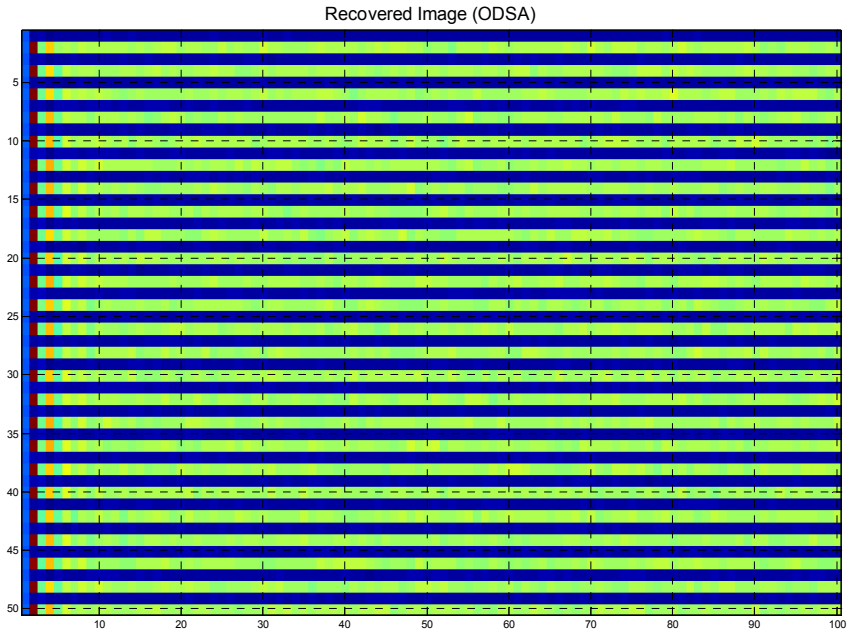

Figure 6-9 The Estimated Image by using the modified version of ODSA

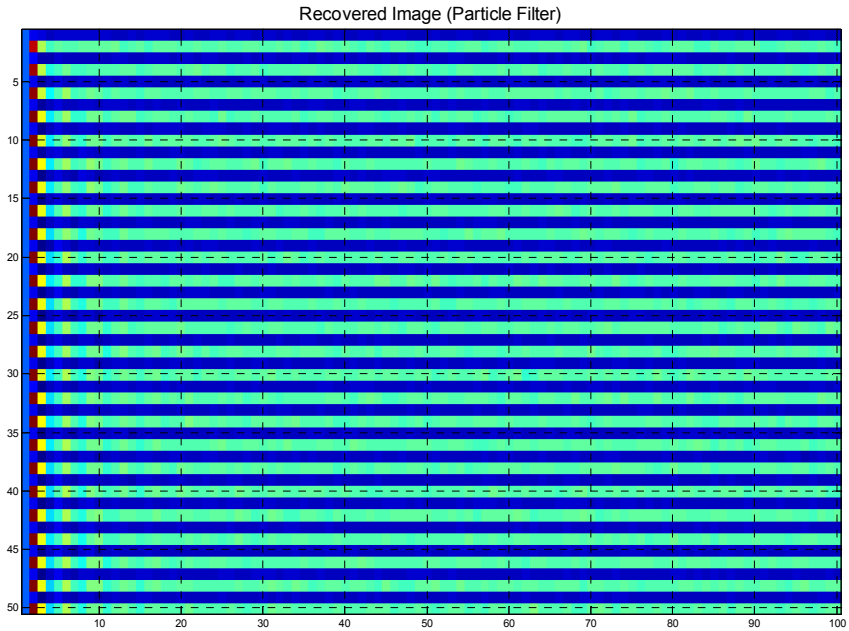

Figure 6-10 The Estimated Image by using the Bootstrap Filter Algorithm

For ODSA estimation, the resulting mean_absolute_error is 6.4217 and the relative error is 27.28%. The time ODSA took for the estimation process is 133.4394 seconds per image.

With the Bootstrap Filter Algorithm the mean_absolute_error became 6.1717 and the relative error was 26.22%. The processing time for the Bootstrap Filter was 124.7213 seconds per image.

6.1.5. Simulation 5

So far, the limit on the number of state nodes was increased to see its effect on the performance of the modified version of ODSA. This time the node limit was kept as the same but the gate size was varied. The simulation parameters for both of the algorithms are as the following:

max_x_k_size	:81
gate_size	:0.01
no_of_sampling_points_x	:3
no_of_sampling_points_w	:3
Ns	:500

The results with these parameters can be seen below:

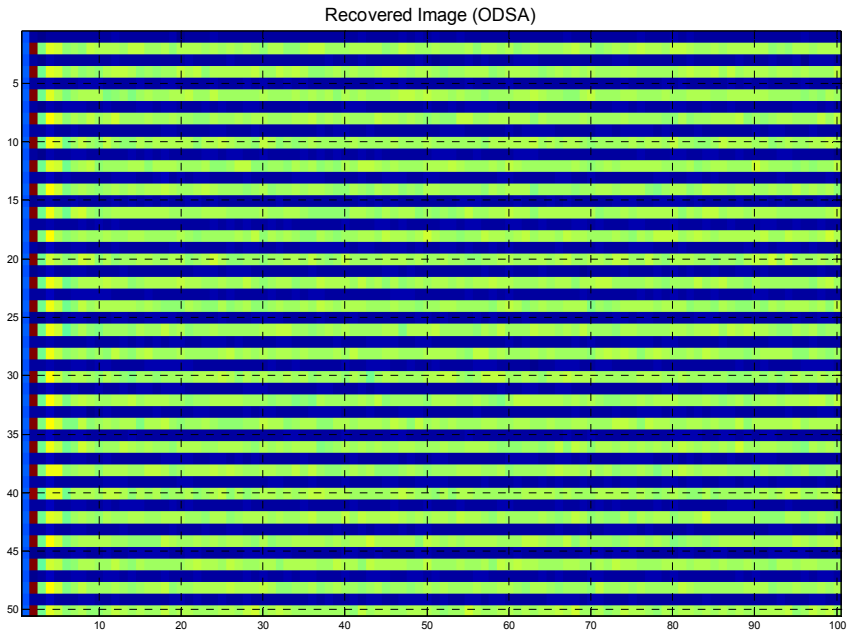

Figure 6-11 The Estimated Image by using the modified version of ODSA

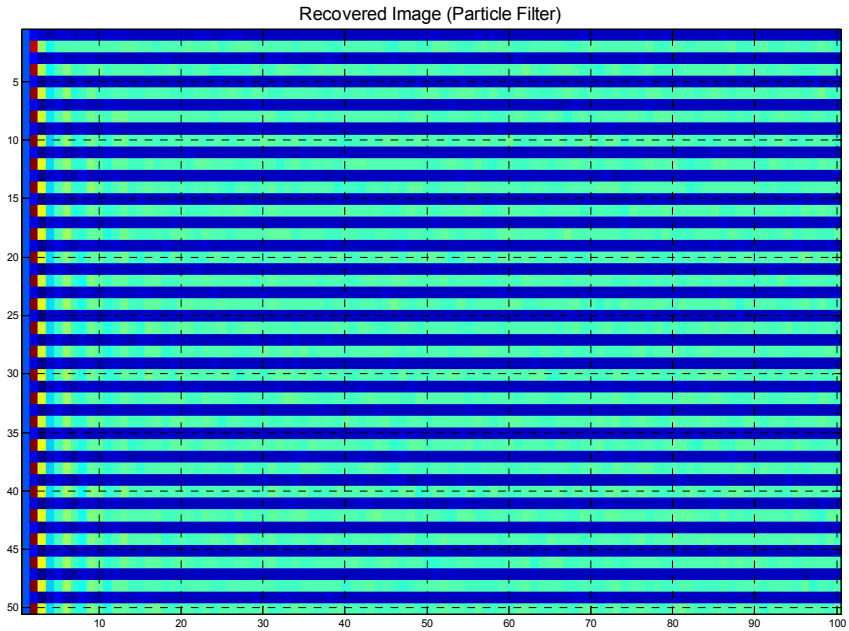

Figure 6-12 The Estimated Image by using the Bootstrap Filter Algorithm

The resulting mean_absolute_error for ODSA estimation is 6.3568 and the relative error is 27.00%. The time ODSA took for the estimation process is 118.0922 seconds per image.

When the Bootstrap Filter Algorithm was performed the mean_absolute_error became 6.0287 and the relative error was 25.60%. The processing time for the Bootstrap Filter was 493.2471 seconds per image.

6.1.6. Simulation 6

This time the following parameters were used for the image restoration process:

max_x_k_size	:81
gate_size	:0.25
no_of_sampling_points_x	:3

no_of_sampling_points_w :3

Ns :750

The results are given below:

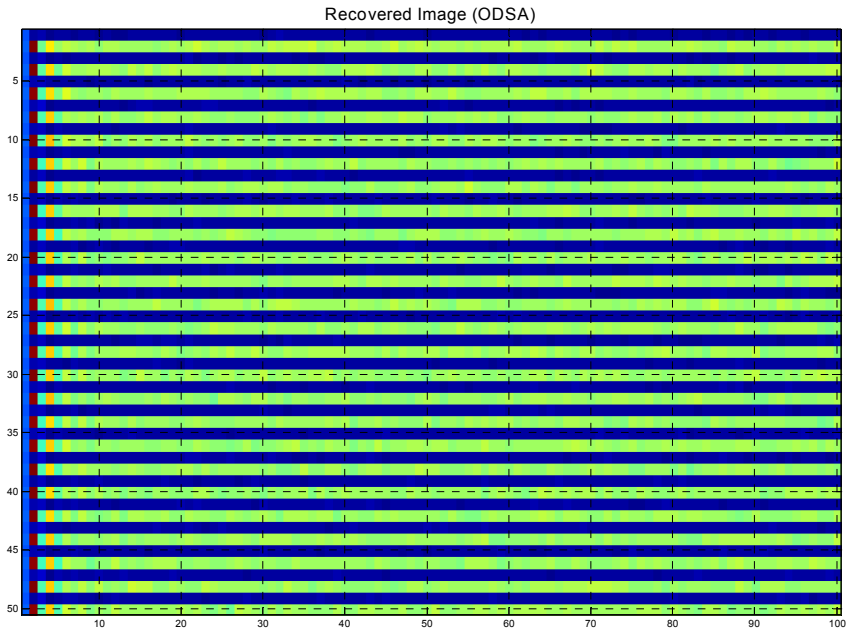

Figure 6-13 The Estimated Image by using the modified version of ODSA

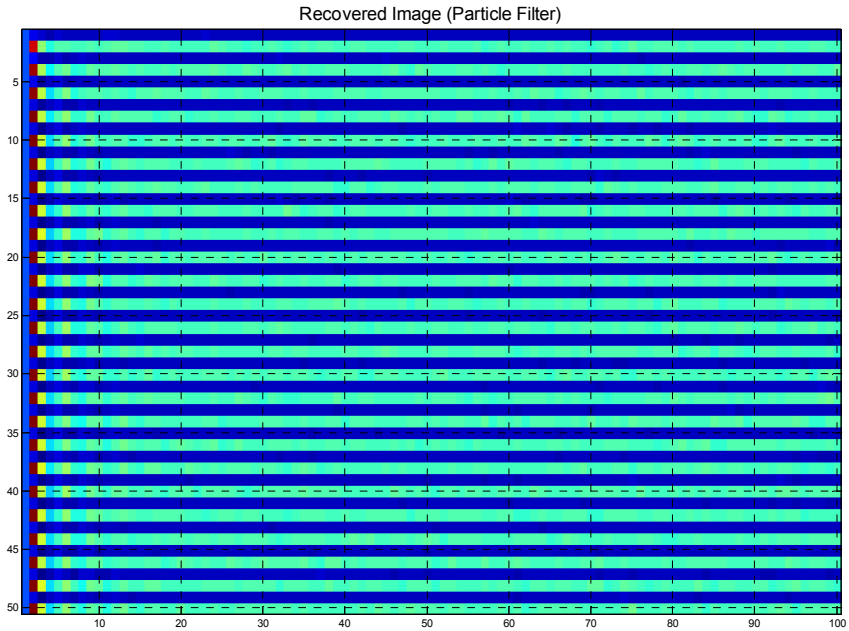

Figure 6-14 The Estimated Image by using the Bootstrap Filter Algorithm

For ODSA estimation, the resulting mean_absolute_error is 6.7759 and the relative error is 28.76%. The time ODSA took for the estimation process is 183.1950 seconds per image.

With the Bootstrap Filter Algorithm the mean_absolute_error became 5.9729 and the relative error was 25.37%. The processing time for the Bootstrap Filter was 1103.3000 seconds per image.

6.1.7. Simulation 7

From this simulation on, the image restoration process was performed only by using the modified version of ODSA. The mentioned simulations were performed with different number of possible values of the initial state variable, x(0) and the disturbance noise variable, w(k). The parameters used are listed below:

max_x_k_size :50

gate_size :0.1

no_of_sampling_points_x :5

no_of_sampling_points_w :5

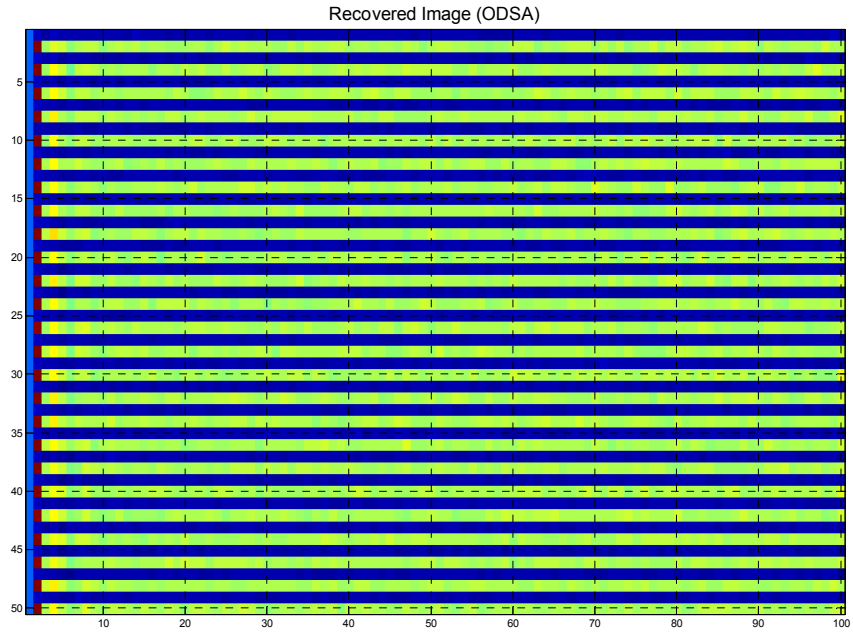

Figure 6-15 The Estimated Image by using the modified version of ODSA

The resulting mean_absolute_error is 6.4275 and the relative error is 27.30%. The time ODSA took for the estimation process is 419.4307 seconds per image in this case.

6.1.8. Simulation 8

The last simulation for this model was performed with the following parameters:

max_x_k_size :50

gate_size :0.1

no_of_sampling_points_x :7

no_of_sampling_points_w :7

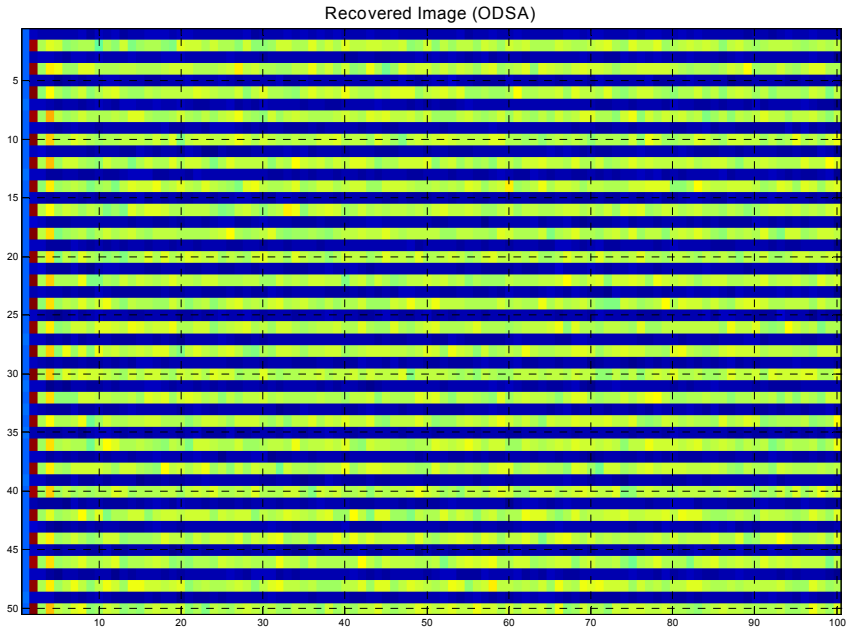

Figure 6-16 The Estimated Image by using the modified version of ODSA

The resulting mean_absolute_error for this case is 6.5142 and the relative error is 27.64%. The time ODSA took for the estimation process is 1158.9250 seconds per image.

6.1.9. Summary of the Results

The graphics showing the error and process time performance of the estimation algorithms with respect to the change in the parameter values are given below:

Timing Performance Graphics:

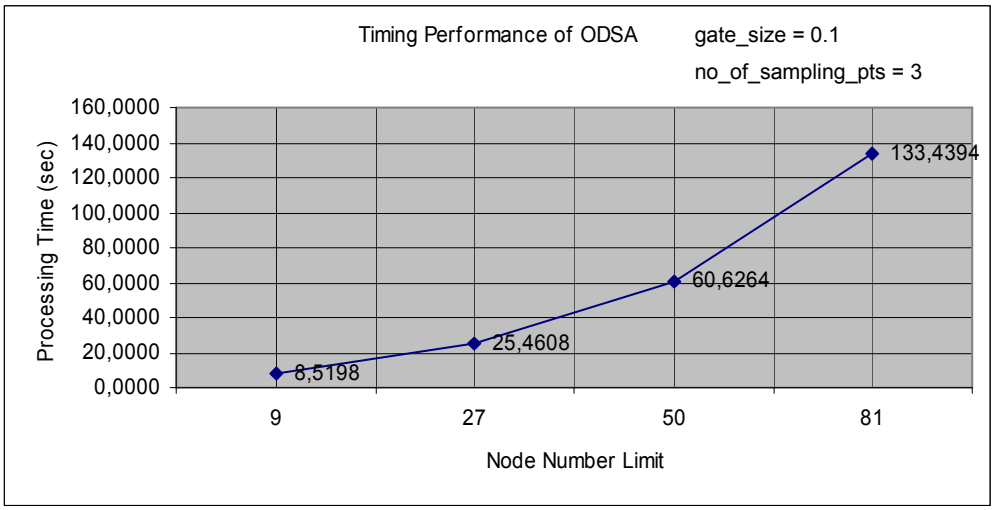

Figure 6-17 Timing Performance of ODSA with the parameters given on the graph

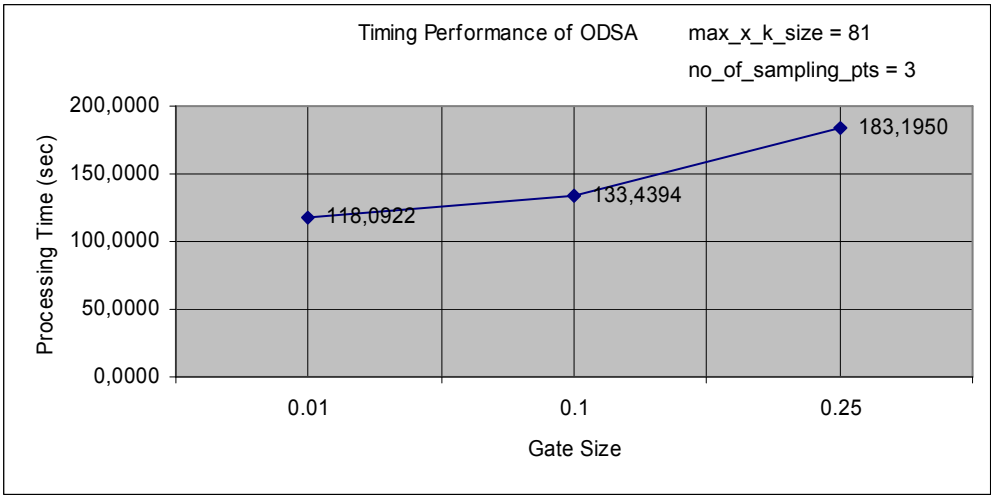

Figure 6-18 Timing Performance of ODSA with the parameters given on the graph

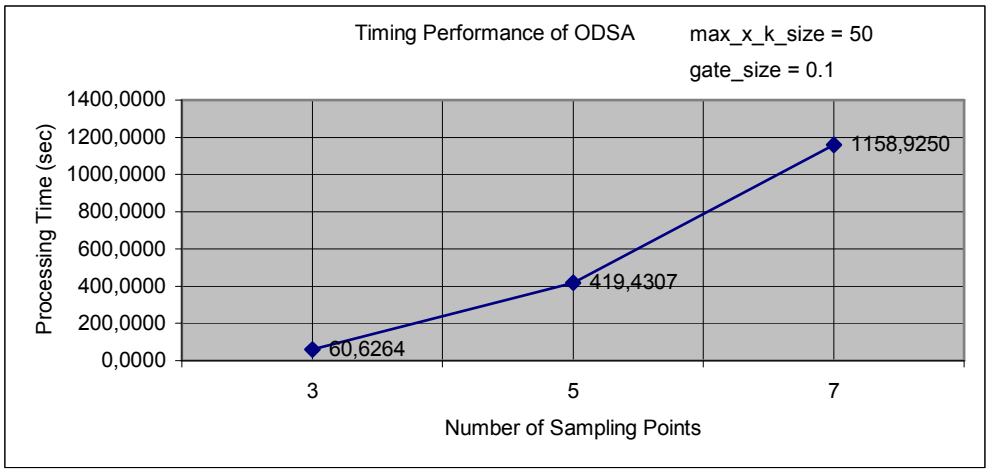

Figure 6-19 Timing Performance of ODSA with the parameters given on the graph

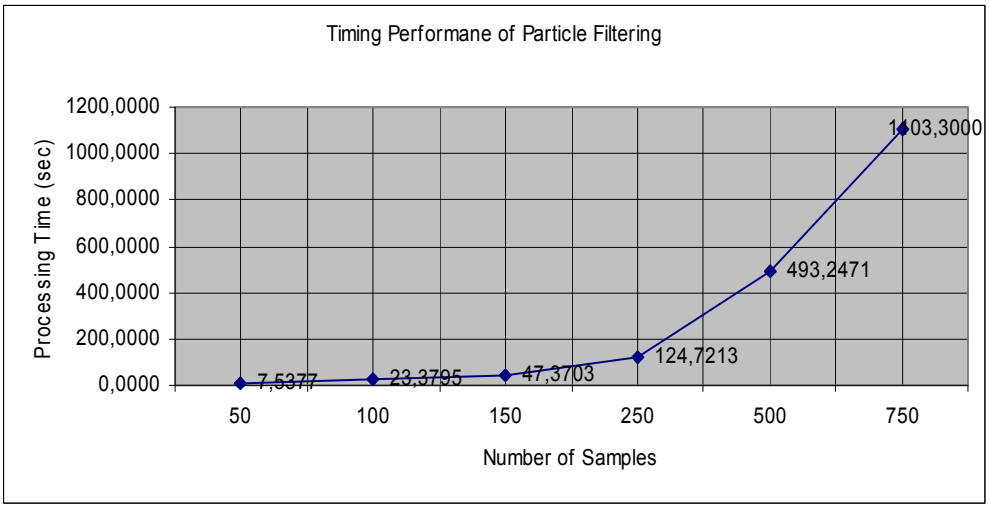

Figure 6-20 Timing Performance of the Bootstrap Filter Algorithm w.r.t. Ns

Error Performance Graphics:

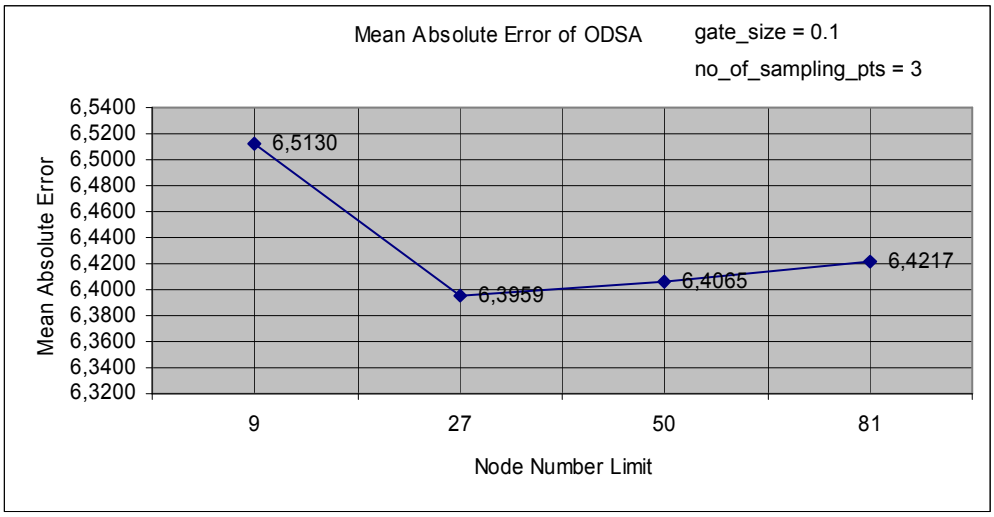

Figure 6-21 Mean absolute error per pixel for ODSA w.r.t. the parameters given on the graph

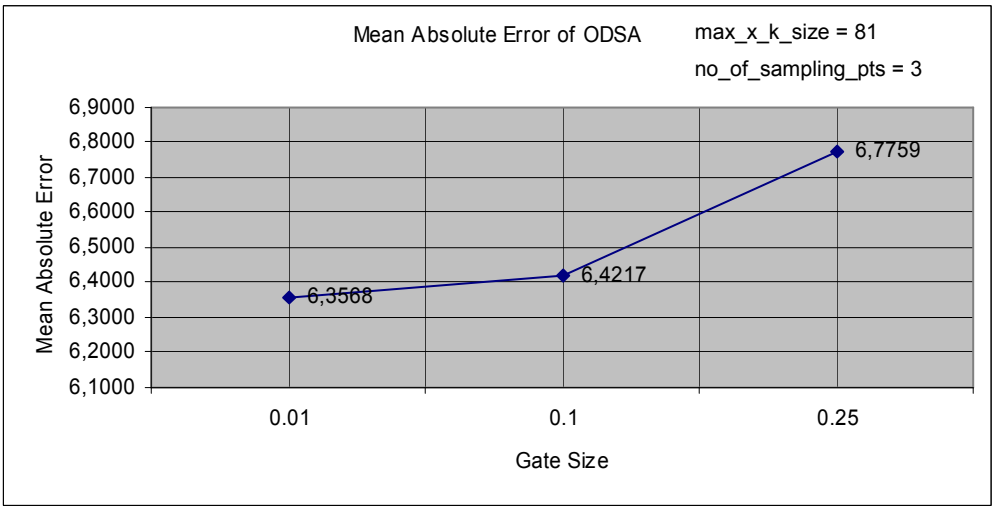

Figure 6-22 Mean absolute error per pixel for ODSA w.r.t. the parameters given on the graph

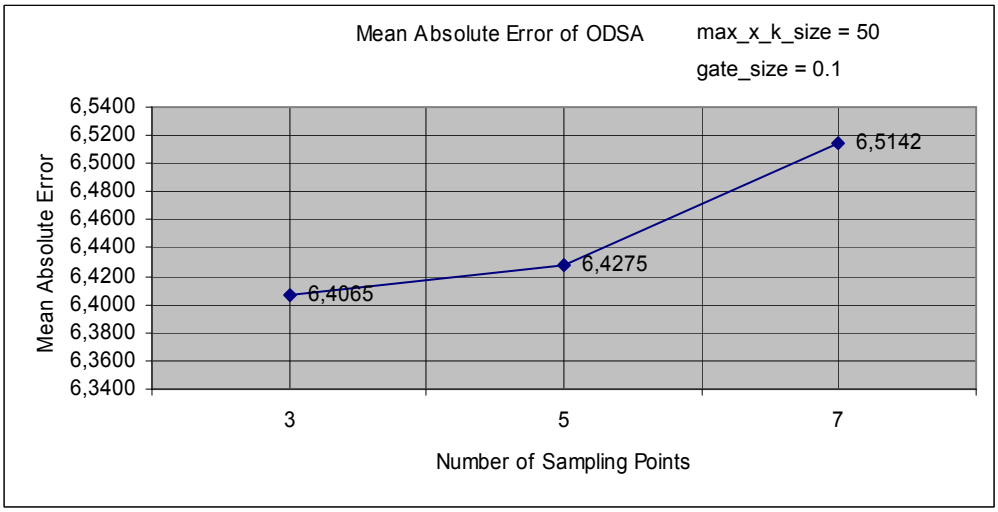

Figure 6-23 Mean absolute error per pixel for ODSA w.r.t. the parameters given on the graph

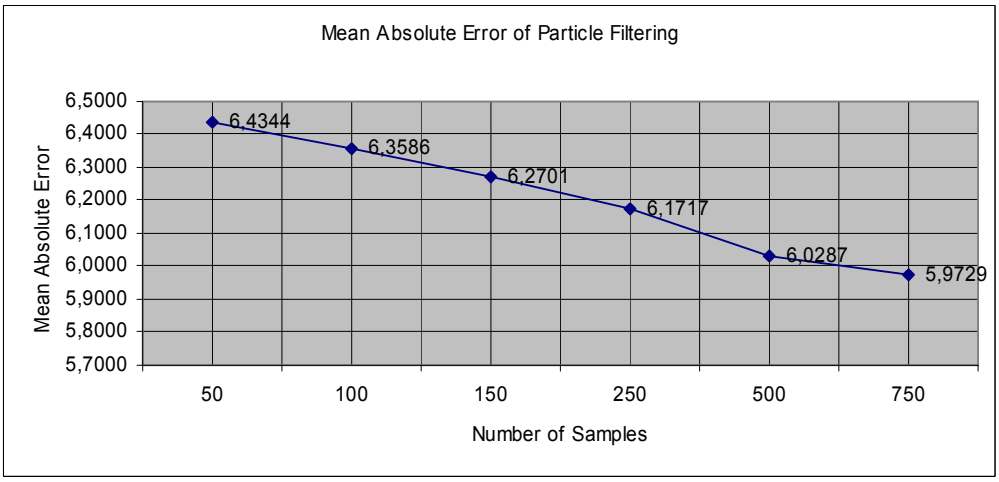

Figure 6-24 Mean absolute error per pixel for the Bootstrap Filter w.r.t Ns

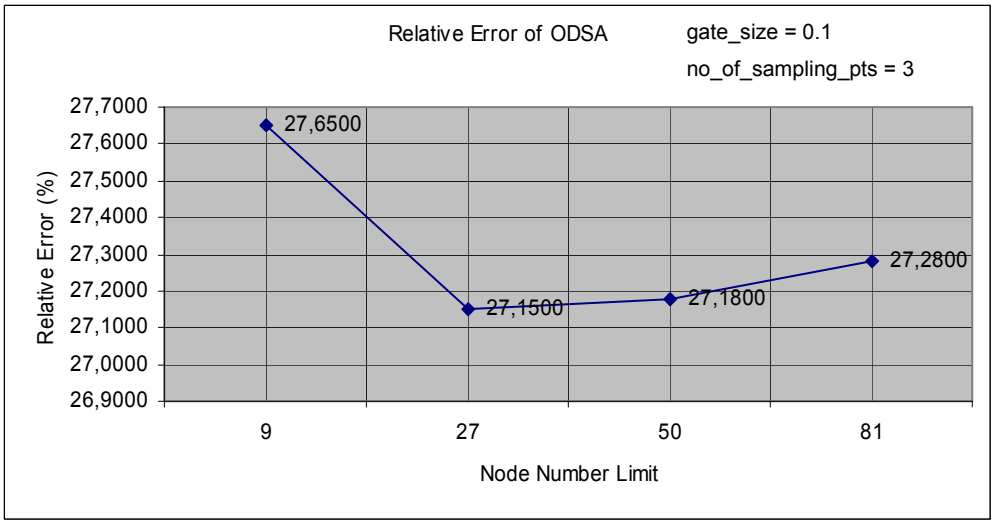

Figure 6-25 Relative error per pixel for ODSA w.r.t. the parameters given on the graph

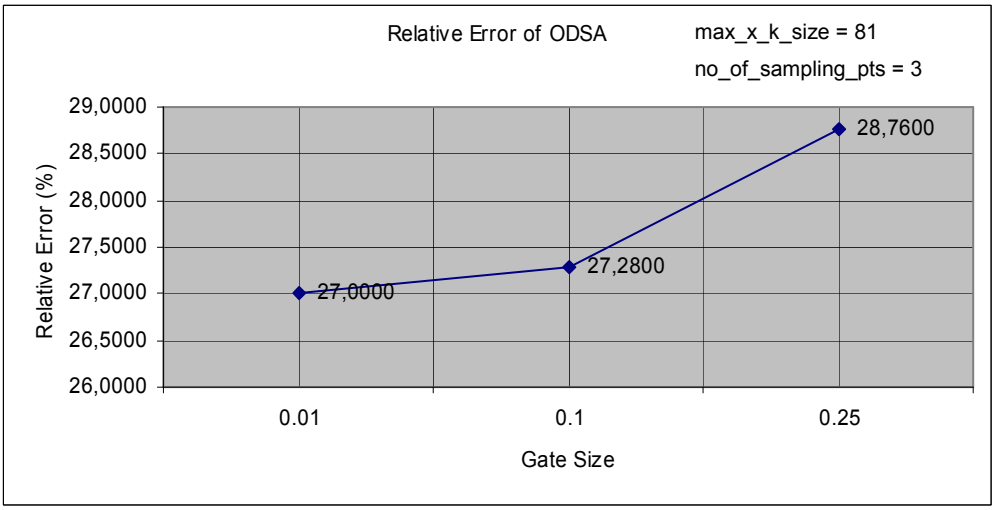

Figure 6-26 Relative error per pixel for ODSA w.r.t. the parameters given on the graph

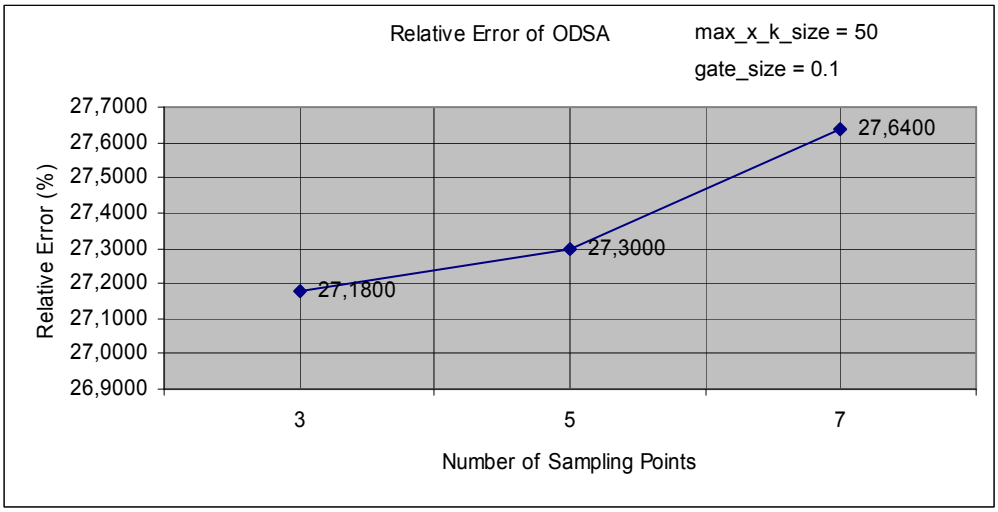

Figure 6-27 Relative error per pixel for ODSA w.r.t. the parameters given on the graph

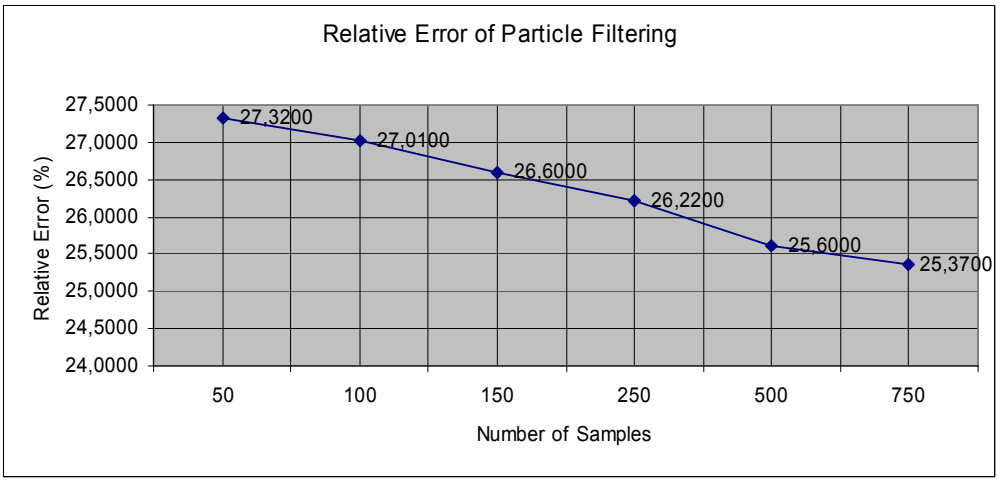

Figure 6-28 Mean absolute error per pixel for the Bootstrap Filter w.r.t Ns

The original image was generated randomly for each of the simulations. Therefore, relative_error value rather than mean_absolute_error should be used for the comparison of error performance of different simulations.

By using ODSA the best estimation for the original image has 27% relative error per pixel. This result was found in 118.0922 seconds for a single image on average. The best estimation obtained with the Bootstrap Filter has 25.37%

relative error per pixel. This result was obtained in 1103.3000 seconds for a single image on average. But, in 47.3703 seconds an estimation with 26.60% relative error per pixel was obtained with 150 samples. So, with the Bootstrap Filter Algorithm [18] a result better than the best one of ODSA can be obtained faster.

The fastest estimation by using ODSA gave 27.65% relative error per pixel in 6.5198 seconds for a single image on average. The fastest estimation with the Bootstrap Filter Algorithm has been reached in 7.6377 seconds for a single image on average with 27.32% relative error per pixel.

6.2. Model 2

The second image to be processed has the following state transition equations:

$$G_1(x_1(k), x_2(k), w_1(k), w_2(k)) = \left| 10\sin^*\left(\frac{\Pi}{2} + \frac{x_2(k)w_2(k)}{4}\right) + \frac{x_1(k)w_1(k)w_2(k)}{4} \right|, \tag{6.12}$$

*: sine of argument in radians

$$G_2(x_1(k), x_2(k), w_1(k), w_2(k)) = \ln\left(\frac{|x_1(k)x_2(k)|}{2} + 10\right) + \frac{|(x_2(k) + w_1(k))w_2(k)|}{2}. \tag{6.13}$$

and the point spread function of the pixels assumed to be:

$$h(x, y, m, n, f(m,n)) = (x+y)(m+n)f^2(m,n). \tag{6.14}$$

For this model the mean of the initial state, x(0) was assumed to be 10 and the variance of it was chosen as 3. The process noise, w(k) and the observation noise v(k) were taken as zero mean Gaussian random variables with variances 1.15 and 1.5 respectively. All the simulations for this model were performed by using MATLAB version 7 running on a computer with an Intel Pentium 4 – 3GHz

processor and 2GB of RAM. For each simulation the estimation process for an image was performed for 250 times and the average of the results were calculated.

The original image and the blurred version of the original image according to the given models and statistics given above are presented in the figures 6-29 and 6-30 respectively.

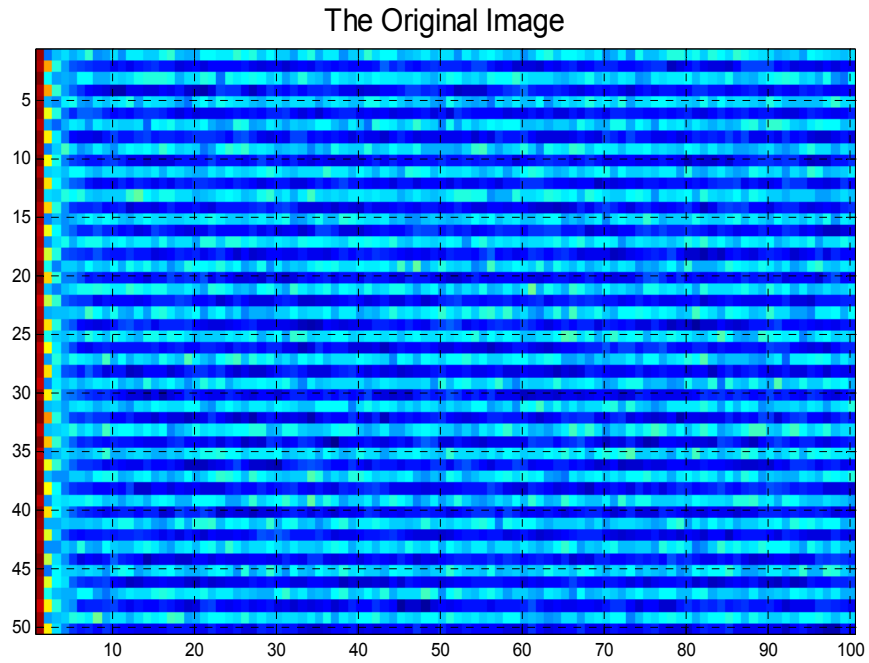

Figure 6-29 The Original Image according to the state transition equation

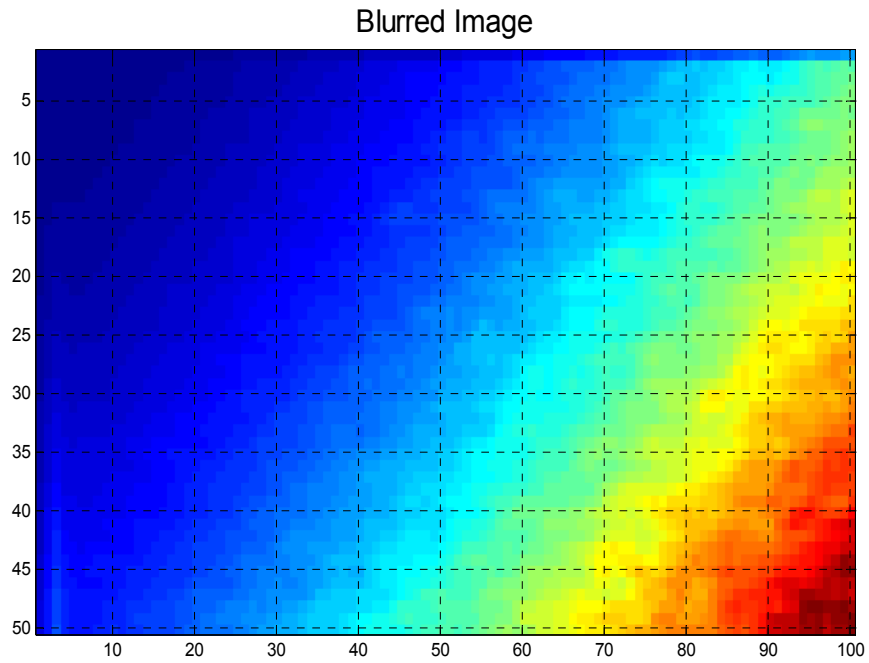

Figure 6-30 Result of the blurring function

6.2.1. Simulation 1

The results of the estimation algorithms with the parameters given below are as the following:

max_x_k_size	:9
gate_size	:0.1
no_of_sampling_points_x	:3
no_of_sampling_points_w	:3
Ns	:50

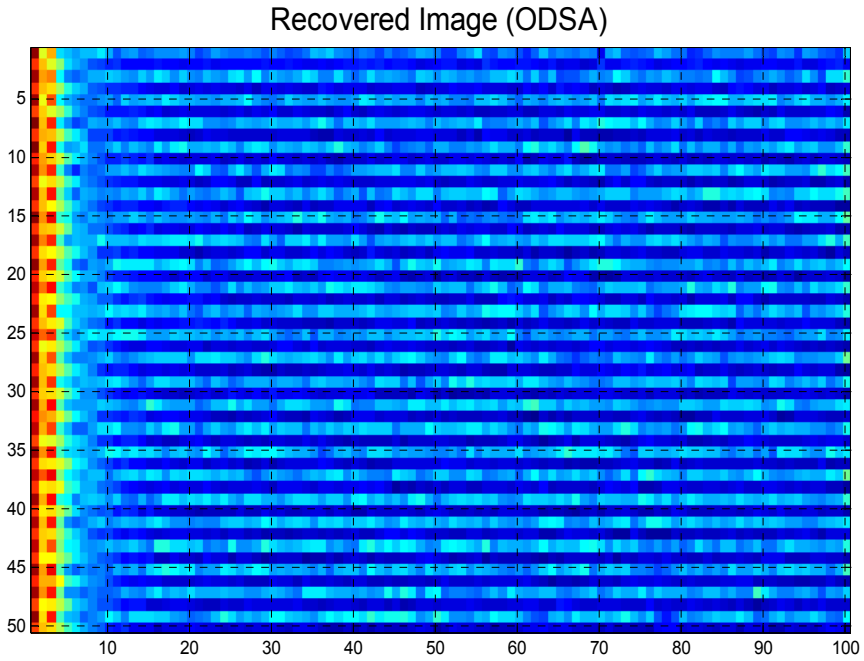

Figure 6-31 The Estimated Image by using the modified version of ODSA

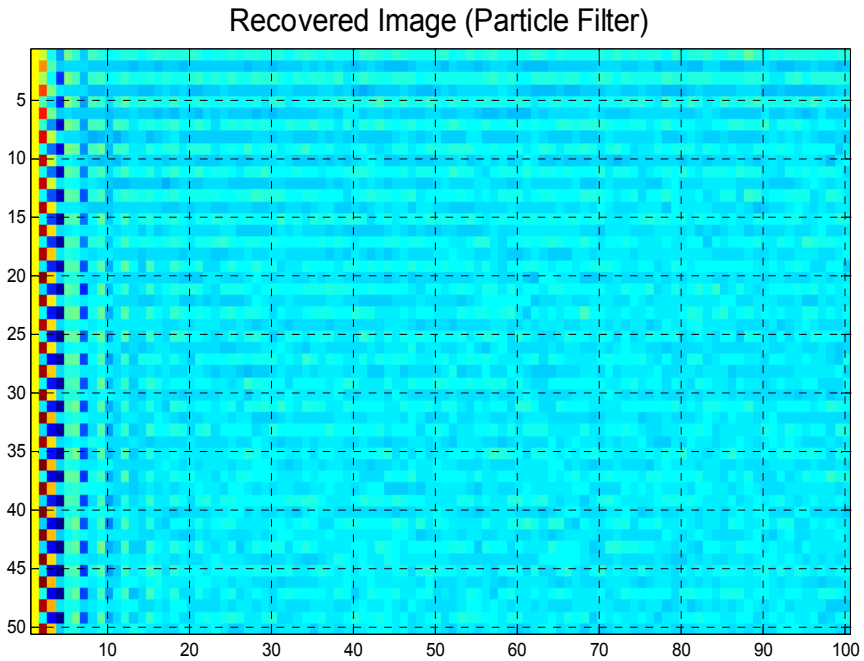

Figure 6-32 The Estimated Image by using the Bootstrap Filter Algorithm

The resulting mean_absolute_error for ODSA estimation is 2.9017 and the relative error is 45.78%. The time ODSA took for the estimation process is 5.5551 seconds per image.

When the Bootstrap Filter Algorithm was performed the mean_absolute_error became 1.6345 and the relative error was 25.78%. The processing time for the Bootstrap Filter was 5.5531 seconds per image.

6.2.2. Simulation 2

The second simulation for this model was performed with the following parameters:

max_x_k_size	:27
gate_size	:0.1
no_of_sampling_points_x	:3
no_of_sampling_points_w	:3
Ns	:100

The estimated images and the performance of the two algorithms are given below:

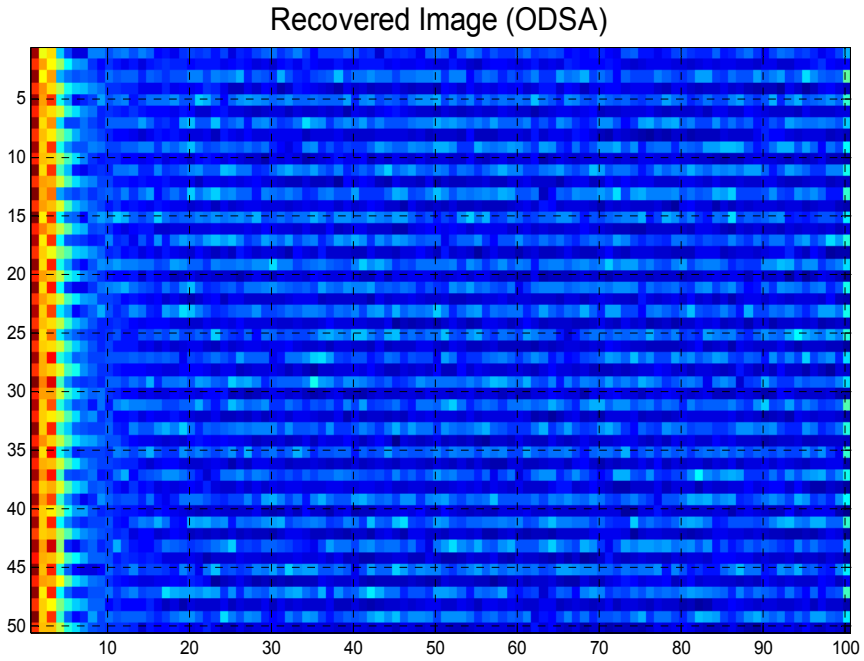

Figure 6-33 The Estimated Image by using the modified version of ODSA

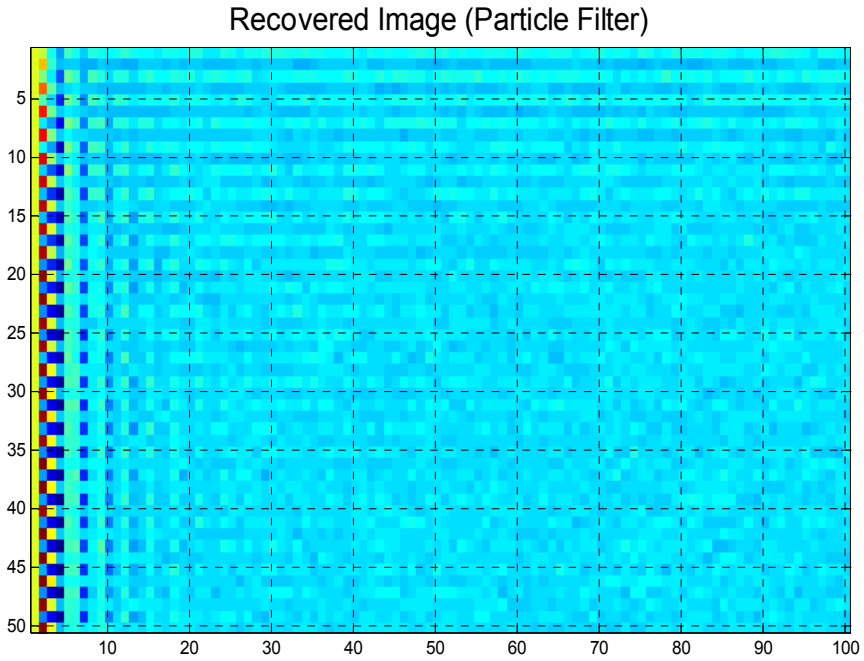

Figure 6-34 The Estimated Image by using the Bootstrap Filter Algorithm

The resulting mean_absolute_error for ODSA estimation is 2.8662 and the relative error is 45.24%. The time ODSA took for the estimation process is 17.7505 seconds per image.

When the Bootstrap Filter Algorithm was performed the mean_absolute_error became 1.5344 and the relative error was 24.21%. The processing time for the Bootstrap Filter was 16.3987 seconds per image.

6.2.3. Simulation 3

This time the parameters were set as the following:

max_x_k_size	:50
gate_size	:0.1
no_of_sampling_points_x	:3
no_of_sampling_points_w	:3
Ns	:150

With the parameters given above, the following results were obtained:

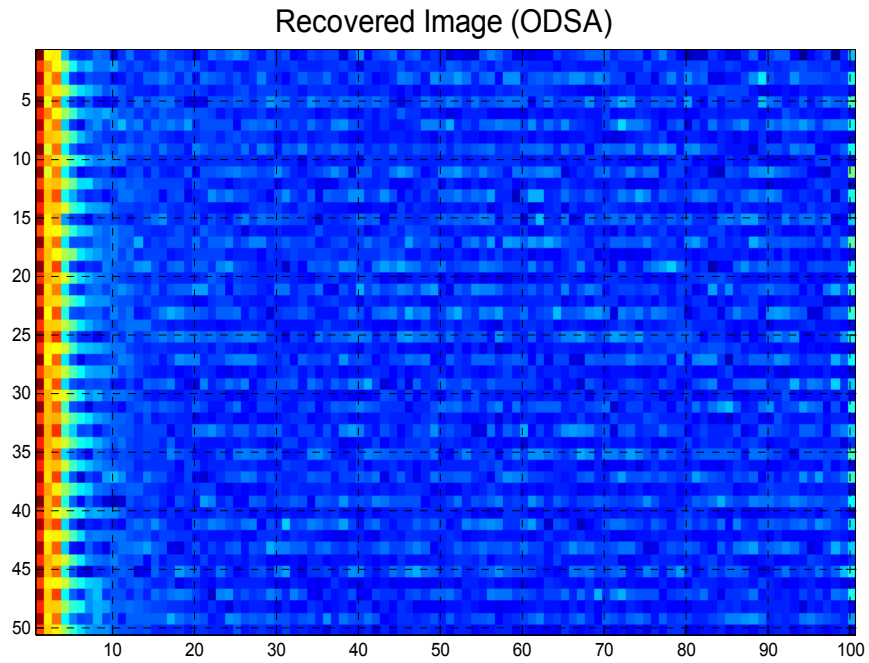

Figure 6-35 The Estimated Image by using the modified version of ODSA

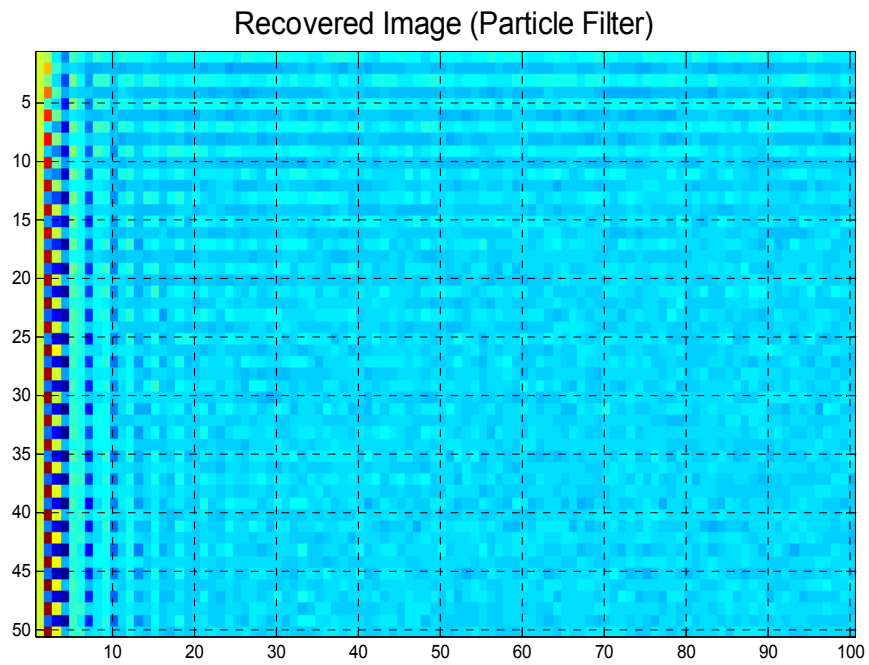

Figure 6-36 The Estimated Image by using the Bootstrap Filter Algorithm

The resulting mean_absolute_error for ODSA estimation is 2.8888 and the relative error is 45.57%. The time ODSA took for the estimation process is 45.6395 seconds per image.

With the Bootstrap Filter Algorithm the mean_absolute_error became 1.4764 and the relative error was 23.31%. The processing time for the Bootstrap Filter was 32.9648 seconds per image.

6.2.4. Simulation 4

For this simulation the parameters were changed to the following values:

max_x_k_size	:81
gate_size	:0.1
no_of_sampling_points_x	:3
no_of_sampling_points_w	:3
Ns	:250

The results are given below:

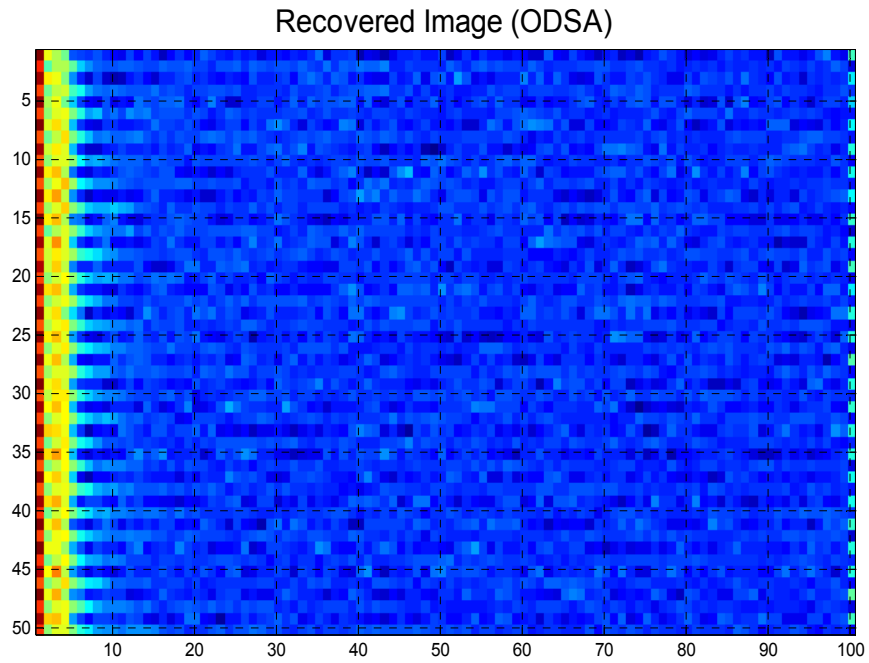

Figure 6-37 The Estimated Image by using the modified version of ODSA

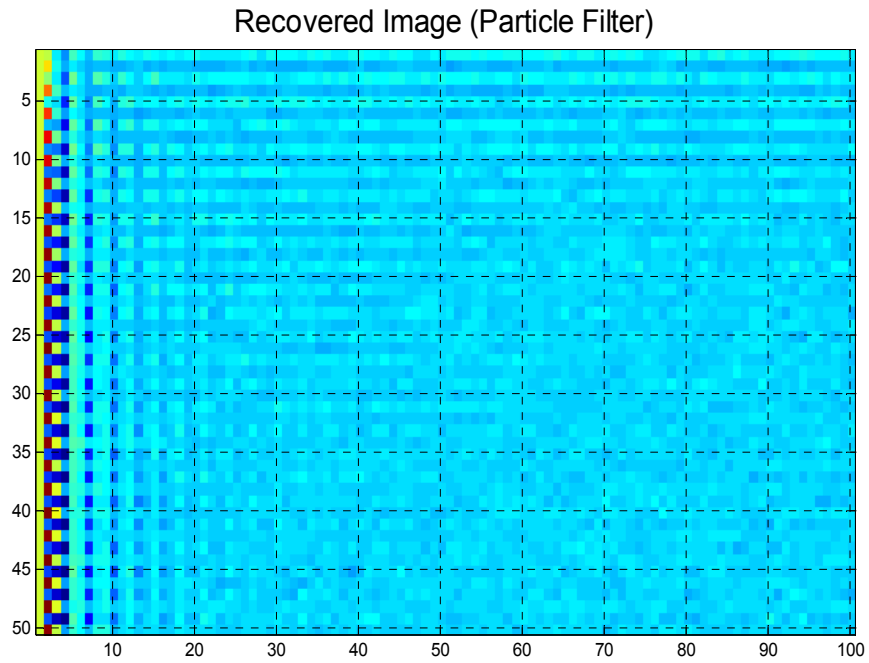

Figure 6-38 The Estimated Image by using the Bootstrap Filter Algorithm

For ODSA estimation, the resulting mean_absolute_error is 2.9221 and the relative error is 46.08%. The time ODSA took for the estimation process is 102.6998 seconds per image.

With the Bootstrap Filter Algorithm the mean_absolute_error became 1.3975 and the relative error was 22.06%. The processing time for the Bootstrap Filter was 84.9205 seconds per image.

6.2.5. Simulation 5

So far, the limit on the number of state nodes was increased to see its effect on the performance of the modified version of ODSA. This time the node limit was kept as the same but the gate size was varied. The simulation parameters for both of the algorithms are as the following:

max_x_k_size	:81
gate_size	:0.01
no_of_sampling_points_x	:3
no_of_sampling_points_w	:3
Ns	:500

The results with these parameters can be seen below:

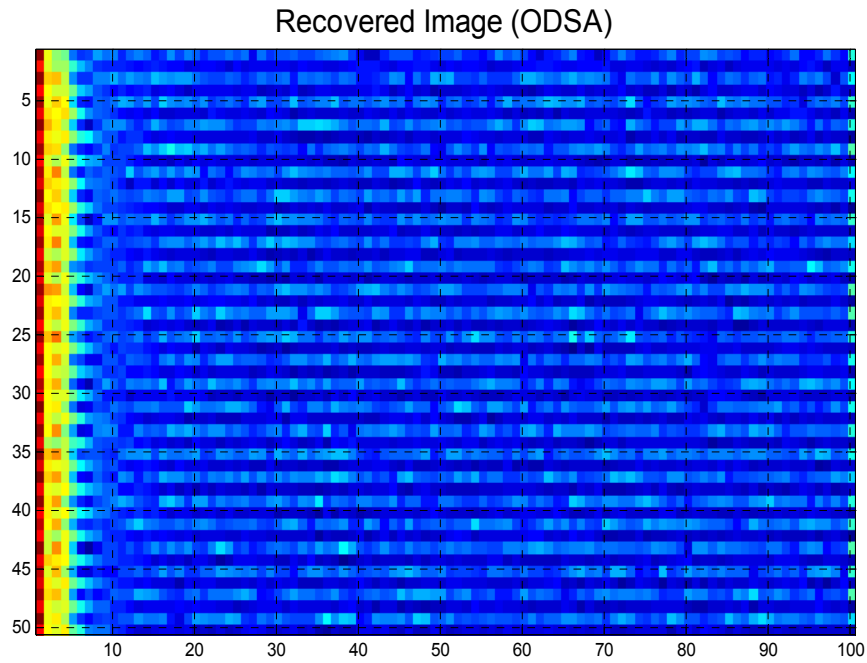

Figure 6-39 The Estimated Image by using the modified version of ODSA

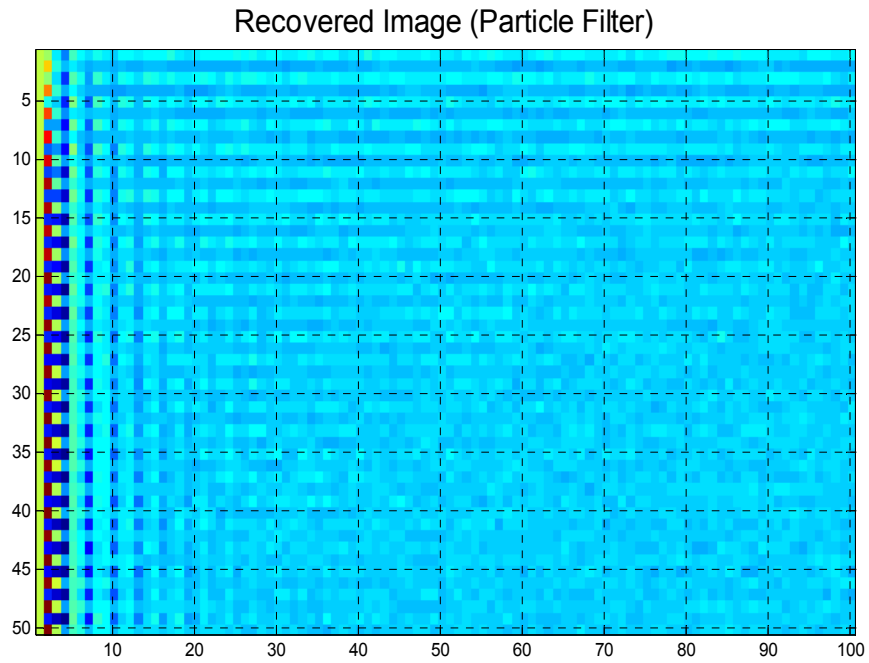

Figure 6-40 The Estimated Image by using the Bootstrap Filter Algorithm

The resulting mean_absolute_error for ODSA estimation is 2.9007 and the relative error is 45.81%. The time ODSA took for the estimation process is 81.2353 seconds per image.

When the Bootstrap Filter Algorithm was performed the mean_absolute_error became 1.3027 and the relative error was 20.55%. The processing time for the Bootstrap Filter was 327.1459 seconds per image.

6.2.6. Simulation 6

This time the following parameters were used for the image restoration process:

max_x_k_size	:81
gate_size	:0.25
no_of_sampling_points_x	:3

no_of_sampling_points_w :3

Ns :750

The results are given below:

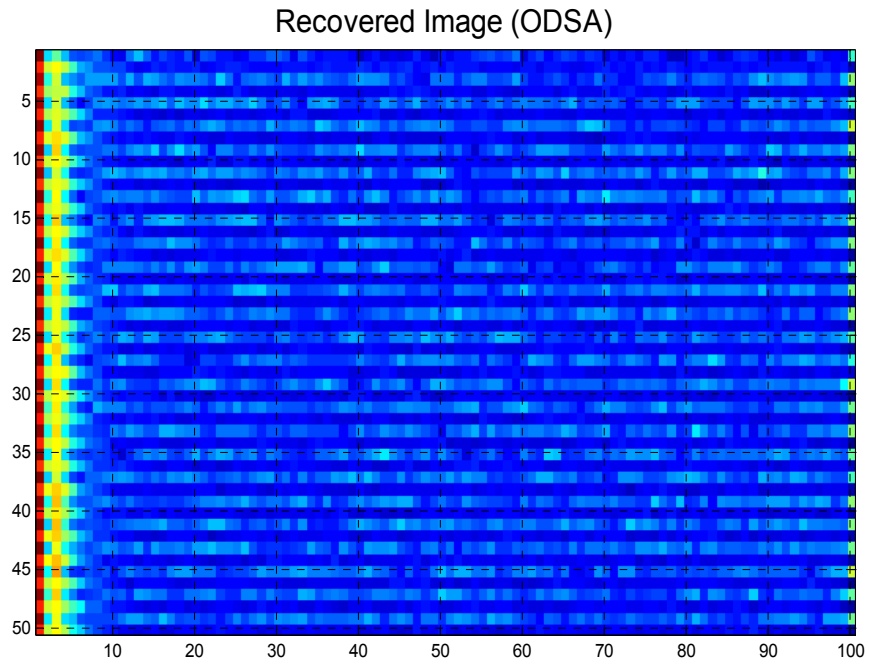

Figure 6-41 The Estimated Image by using the modified version of ODSA

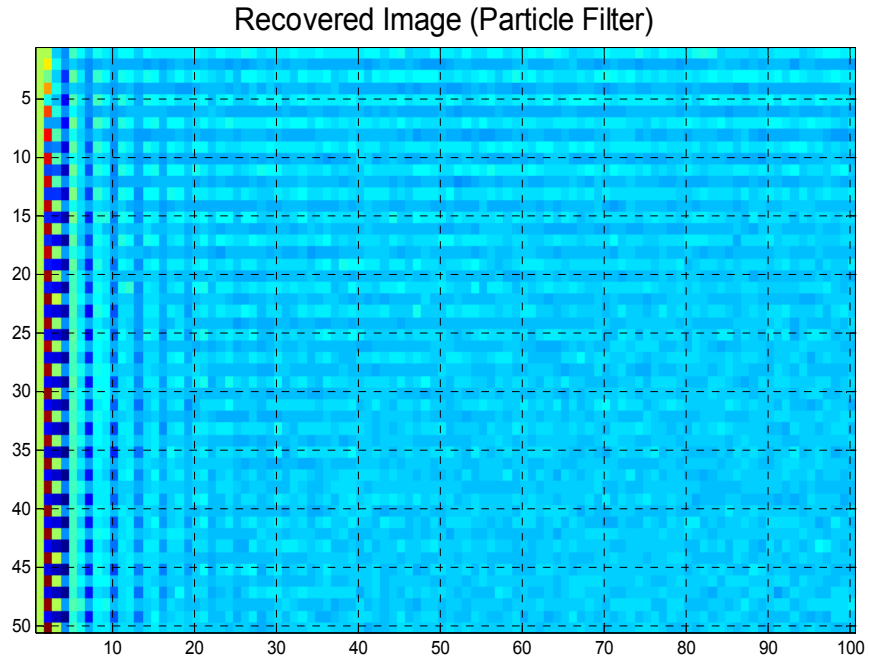

Figure 6-42 The Estimated Image by using the Bootstrap Filter Algorithm

For ODSA estimation, the resulting mean_absolute_error is 2.8847 and the relative error is 45.52%. The time ODSA took for the estimation process is 144.3357 seconds per image.

With the Bootstrap Filter Algorithm the mean_absolute_error became 1.2356 and the relative error was 19.49%. The processing time for the Bootstrap Filter was 727.9762 seconds per image.

6.2.7. Simulation 7

From this simulation on, the image restoration process was performed only by using the modified version of ODSA. The mentioned simulations were performed with different number of possible values of the initial state variable, x(0) and the disturbance noise variable, w(k). The parameters used are listed below:

max_x_k_size :50

gate_size :0.1

no_of_sampling_points_x :5

no_of_sampling_points_w :5

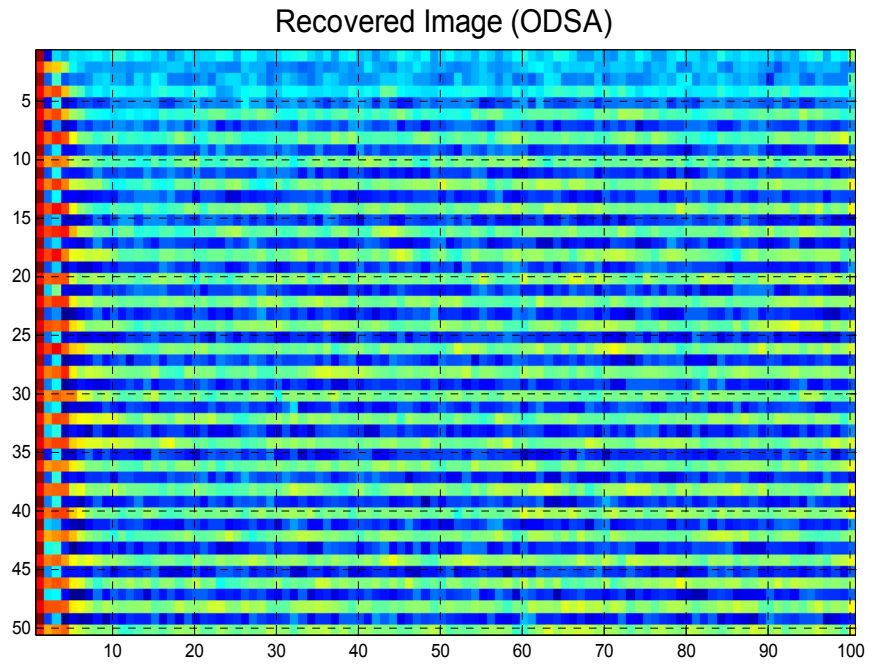

Figure 6-43 The Estimated Image by using the modified version of ODSA

The resulting mean_absolute_error is 3.0115 and the relative error is 47.54%. The time ODSA took for the estimation process is 239.4328 seconds per image in this case.

6.2.8. Simulation 8

The last simulation for this model was performed with the following parameters:

max_x_k_size :50

gate_size :0.1

no_of_sampling_points_x :7

no_of_sampling_points_w :7

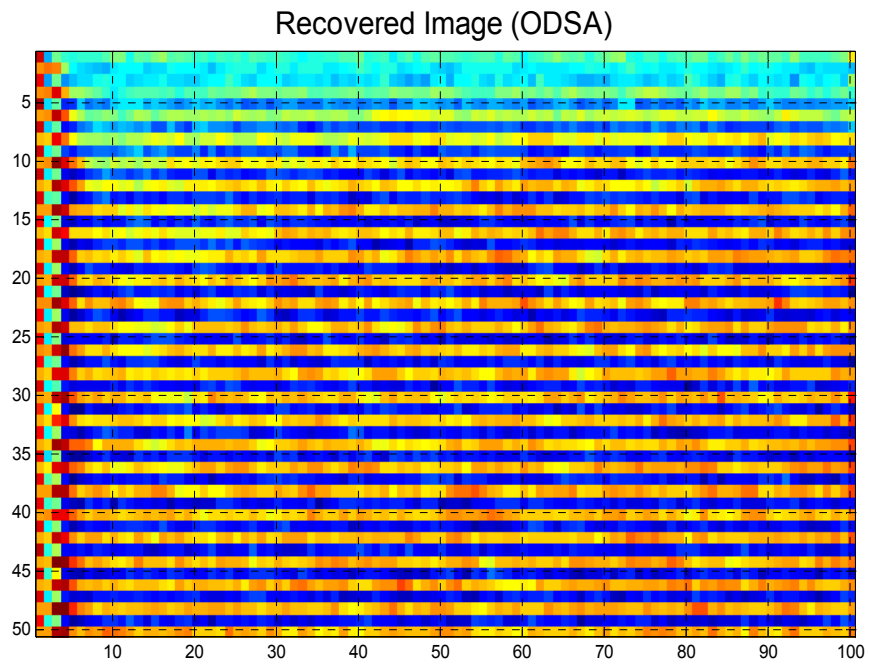

Figure 6-44 The Estimated Image by using the modified version of ODSA

The resulting mean_absolute_error for this case is 3.4717 and the relative error is 54.80%. The time ODSA took for the estimation process is 746.4990 seconds per image.

6.2.9. Summary of the Results

The graphics showing the error and process time performance of the estimation algorithms with respect to the change in the parameter values are given below:

Timing Performance Graphics:

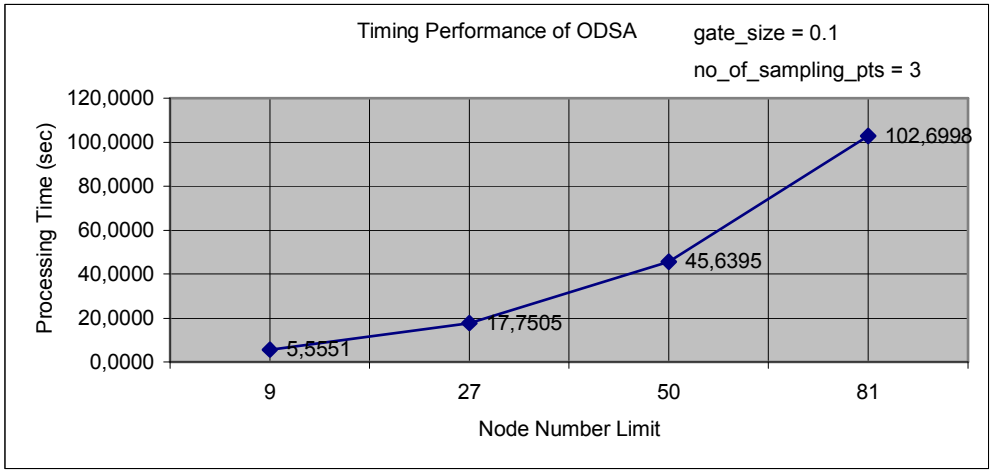

Figure 6-45 Timing Performance of ODSA with the parameters given on the graph

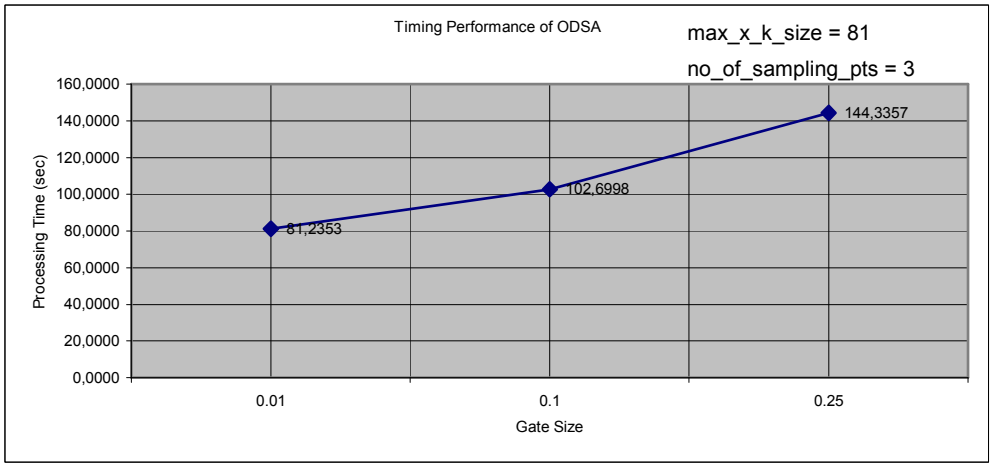

Figure 6-46 Timing Performance of ODSA with the parameters given on the graph

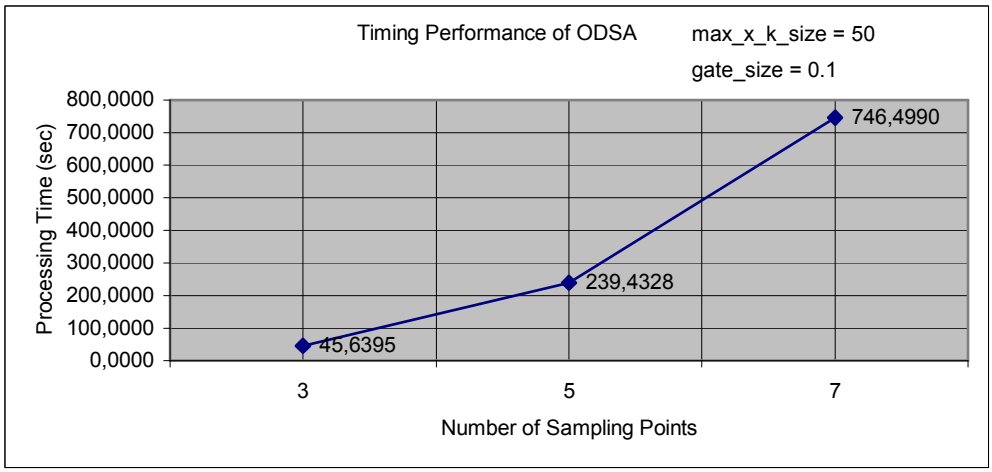

Figure 6-47 Timing Performance of ODSA with the parameters given on the graph

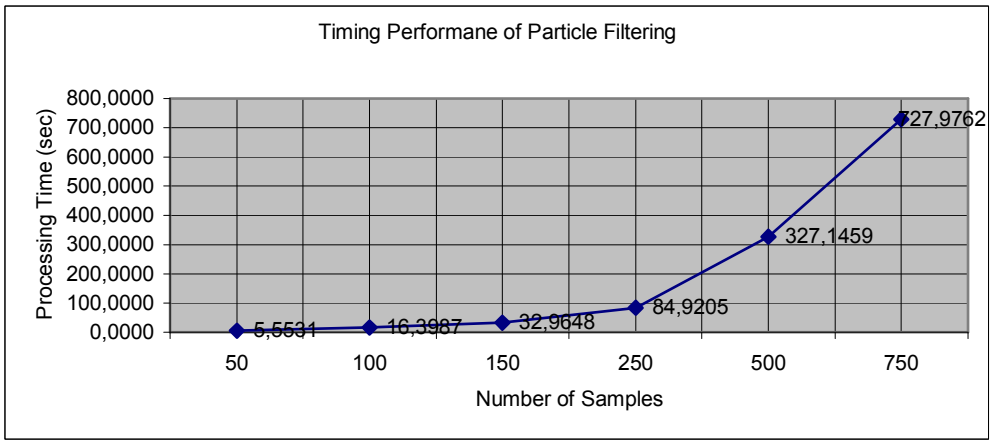

Figure 6-48 Timing Performance of the Bootstrap Filter Algorithm w.r.t. Ns

Error Performance Graphics:

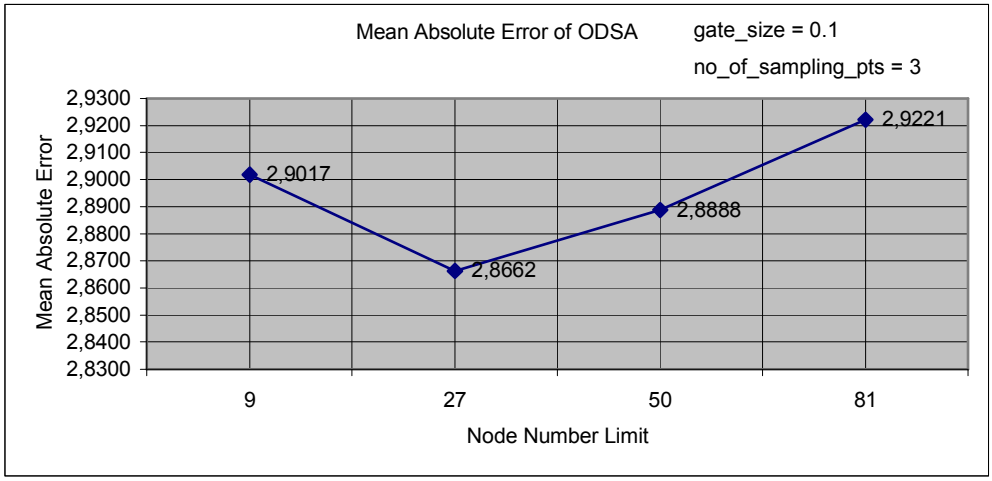

Figure 6-49 Mean absolute error per pixel for ODSA w.r.t. the parameters given on the graph

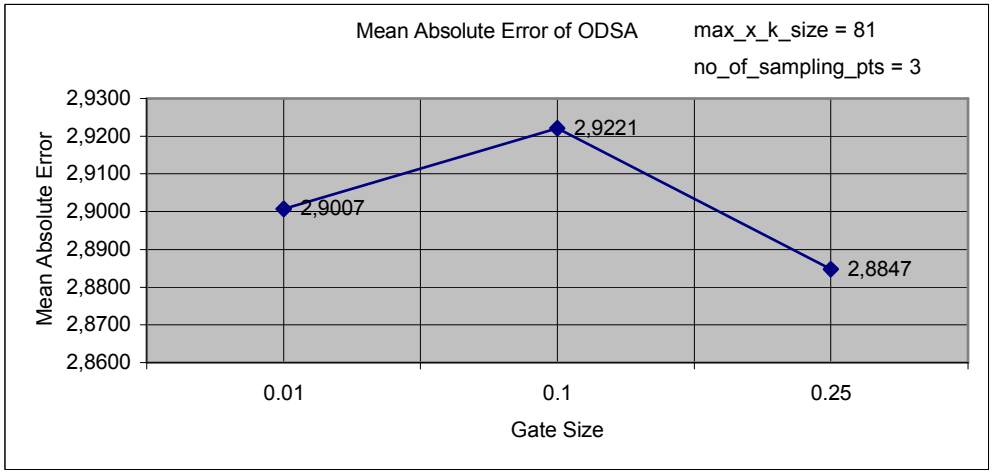

Figure 6-50 Mean absolute error per pixel for ODSA w.r.t. the parameters given on the graph

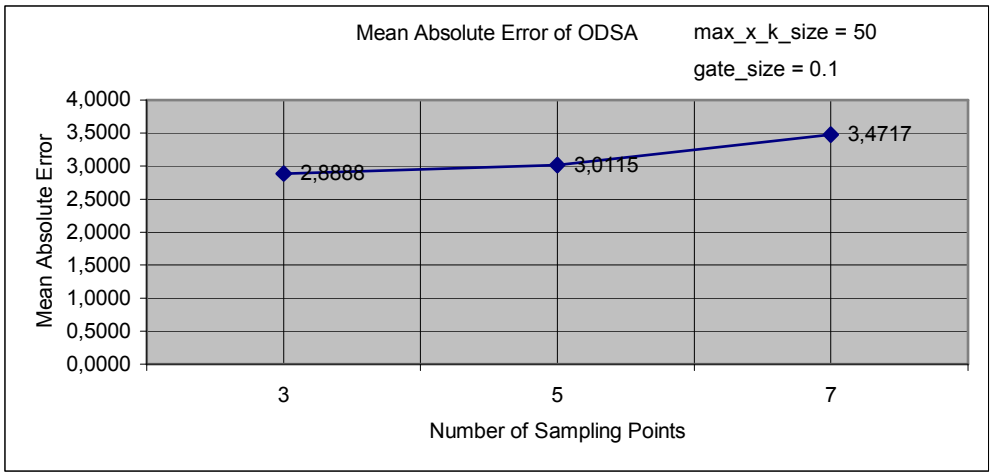

Figure 6-51 Mean absolute error per pixel for ODSA w.r.t. the parameters given on the graph

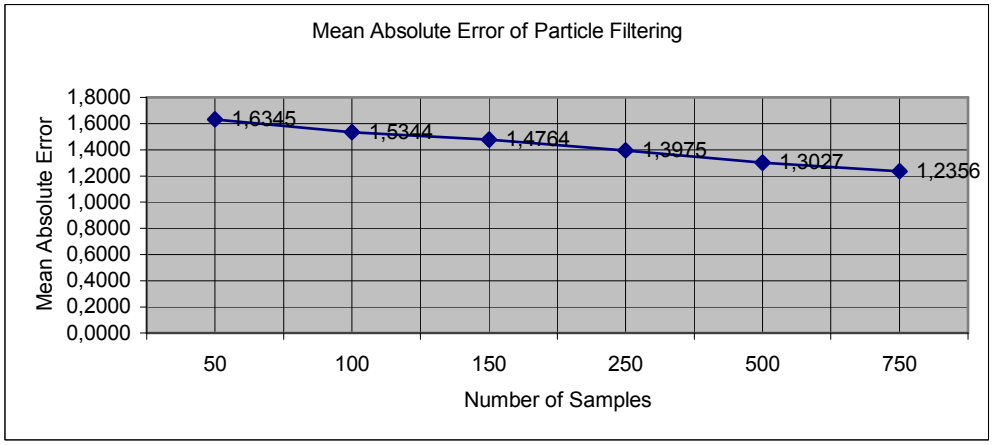

Figure 6-52 Mean absolute error per pixel for the Bootstrap Filter w.r.t Ns

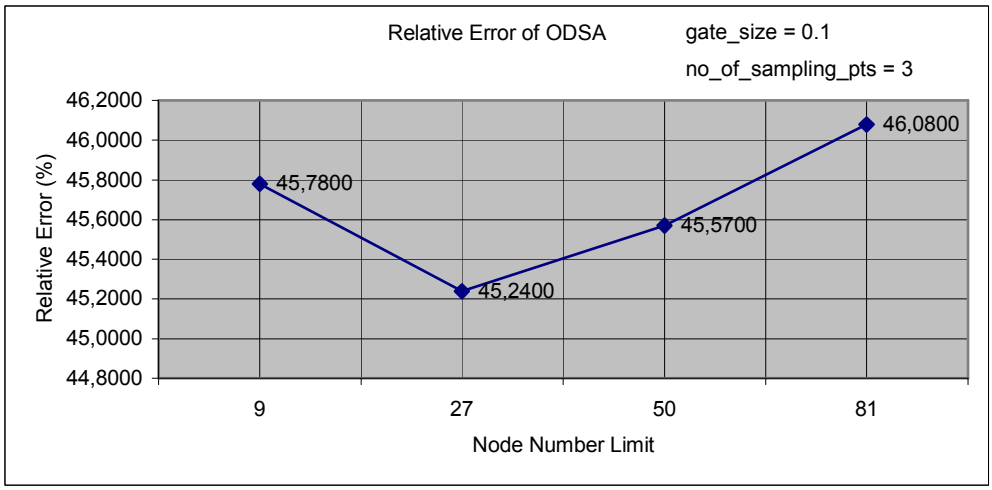

Figure 6-53 Relative error per pixel for ODSA w.r.t. the parameters given on the graph

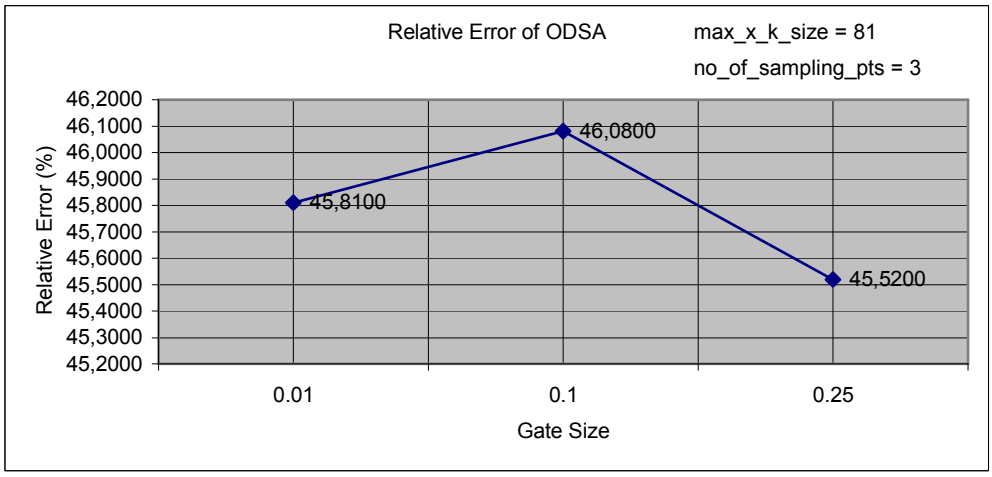

Figure 6-54 Relative error per pixel for ODSA w.r.t. the parameters given on the graph

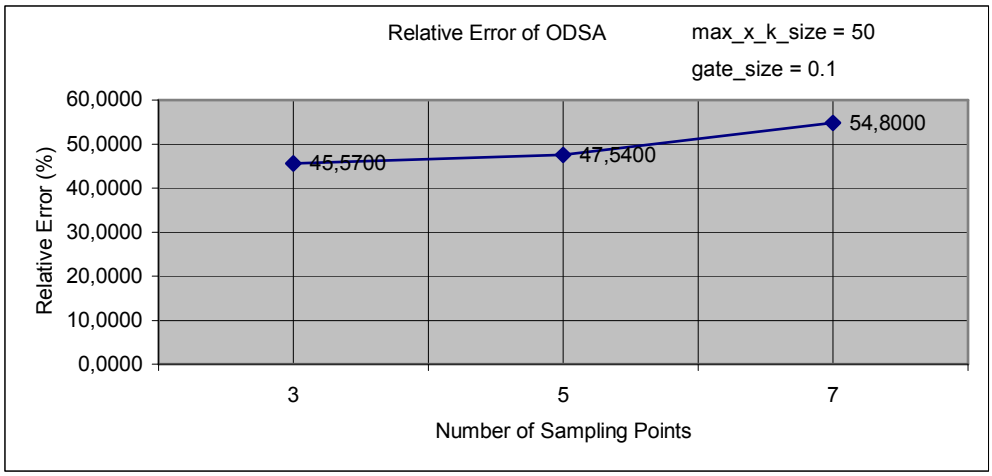

Figure 6-55 Relative error per pixel for ODSA w.r.t. the parameters given on the graph

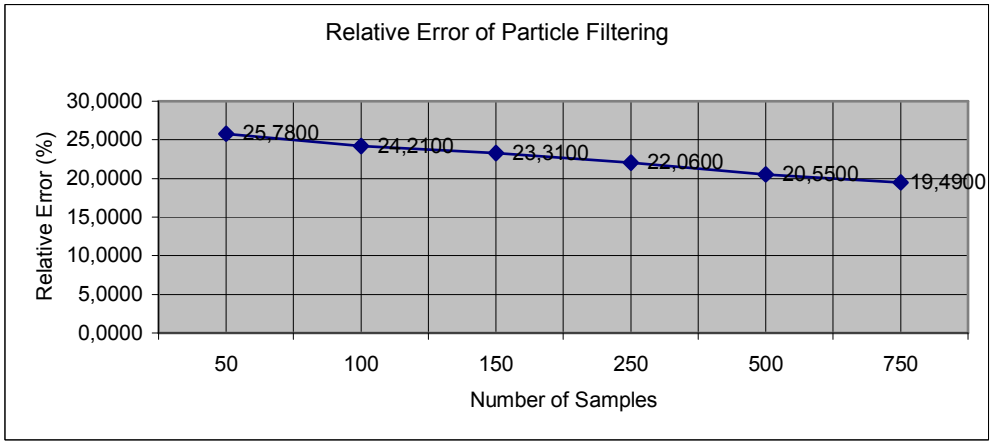

Figure 6-56 Mean absolute error per pixel for the Bootstrap Filter w.r.t Ns

The original image was generated randomly for each of the simulations. Therefore, relative_error value rather than mean_absolute_error should be used for the comparison of error performance of different simulations.

By using ODSA the best estimation for the original image has 45.24% relative error per pixel. This result was found in 17.7505 seconds for a single image on average. The result obtained with the best timing performance has 45.78% relative error per pixel in 5.5551 seconds for a single image on average. The best estimation obtained with the Bootstrap Filter has 19.49%

relative error per pixel. This result was obtained in 727.9762 seconds for a single image on average. The best timing performance of Bootstrap Filter Algorithm was obtained with 25.78% relative error per pixel in 5.5531 seconds per image.

6.3. Model 3

The third image to be processed has the following state transition equations:

$$G_1(x_1(k), x_2(k), w_1(k), w_2(k)) = \sin^*\left(2\Pi w_2(k)\left(x_1^2(k) + x_2^2(k)\right)\right)^2 + \left|x_2(k) + w_1(k)\right| \tag{6.15}$$

$$G_2(x_1(k), x_2(k), w_1(k), w_2(k)) = \left|x_1(k) - x_2(k)\right|^{\frac{2}{3}} + \left|0.95x_1(k) + w_1(k)w_2(k)\right|. \tag{6.16}$$

and the point spread function of the pixels assumed to be:

$$h(x, y, m, n, f(m,n)) = \sin^{*2}(x + y) + \left((m + n)f(m,n)\right)^2. \tag{6.17}$$

*: sine of argument in radians

For this model the mean of the initial state, x(0) was assumed to be 25 and the variance of it was chosen as 3. The process noise, w(k) and the observation noise v(k) were taken as zero mean Gaussian random variables with variances 4 and 2 respectively. All the simulations for this model were performed by using MATLAB version 7 running on a computer with an Intel Xeon – 3.4GHz processor and 4GB of RAM. For each simulation the estimation process for an image was performed for 250 times and the average of the results were calculated.

The original image and the blurred version of the original image according to the given models and statistics given above are presented in the figures 6-57 and 6-58 respectively.

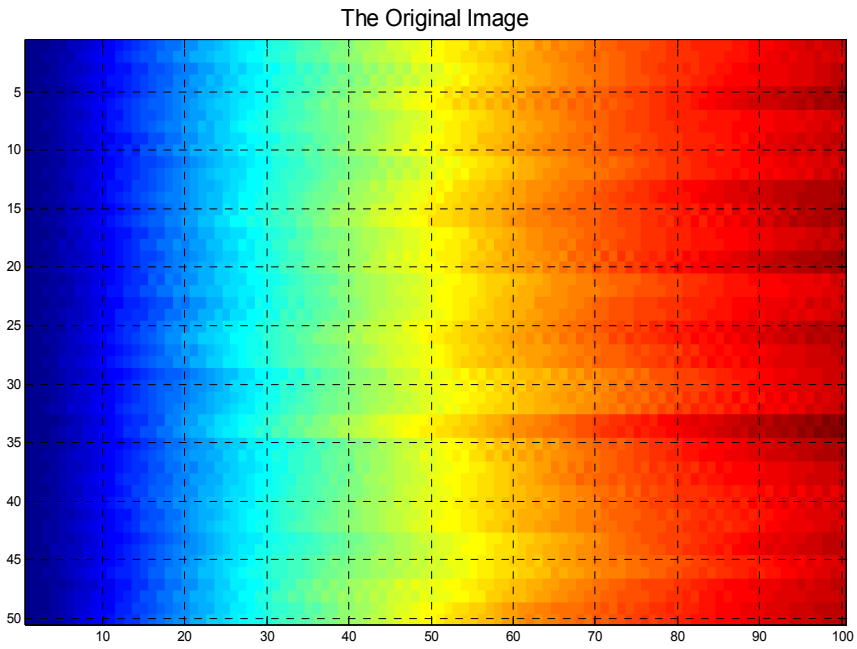

Figure 6-57 The Original Image according to the state transition equation

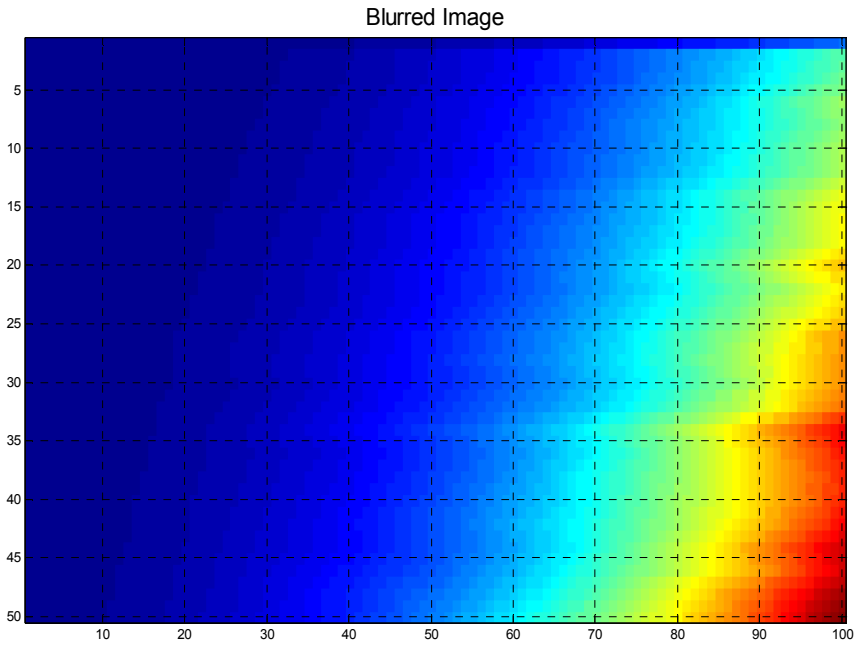

Figure 6-58 Result of the blurring function

6.3.1. Simulation 1

The results of the estimation algorithms with the parameters given below are as the following:

max_x_k_size	:9
gate_size	:0.1
no_of_sampling_points_x	:3
no_of_sampling_points_w	:3
Ns	:50

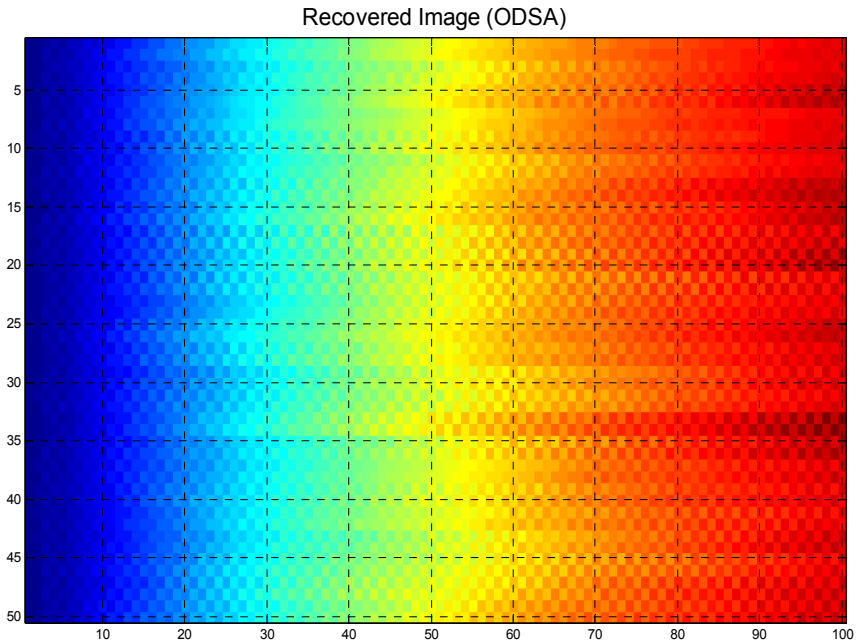

Figure 6-59 The Estimated Image by using the modified version of ODSA

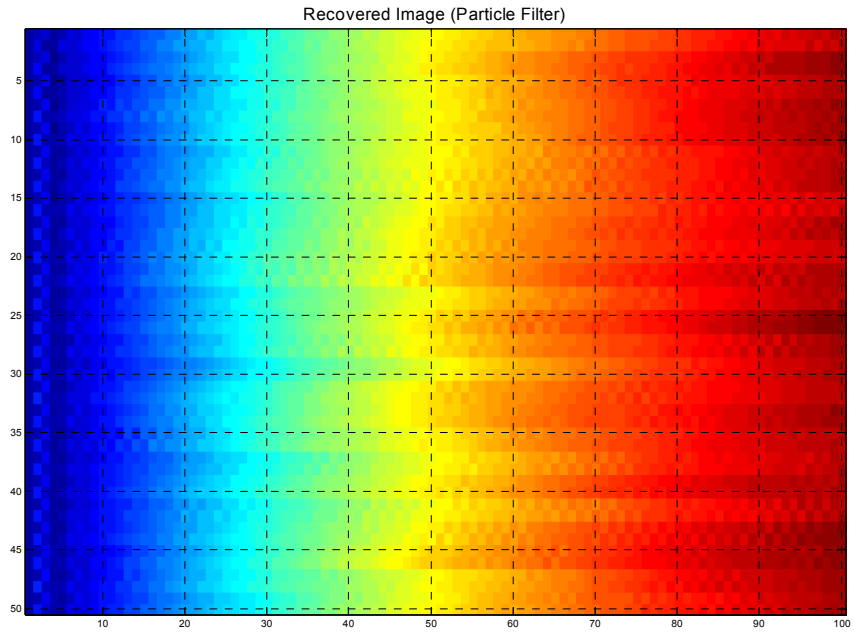

Figure 6-60 The Estimated Image by using the Bootstrap Filter Algorithm

The resulting mean_absolute_error for ODSA estimation is 8.8213 and the relative error is 11.60%. The time ODSA took for the estimation process is 4.9245 seconds per image.

When the Bootstrap Filter Algorithm was performed the mean_absolute_error became 4.9323 and the relative error was 6.57%. The processing time for the Bootstrap Filter was 4.9053 seconds per image.

6.3.2. Simulation 2

The second simulation for this model was performed with the following parameters:

max_x_k_size	:27
gate_size	:0.1
no_of_sampling_points_x	:3

no_of_sampling_points_w :3

Ns :100

The estimated images and the performance of the two algorithms are given below:

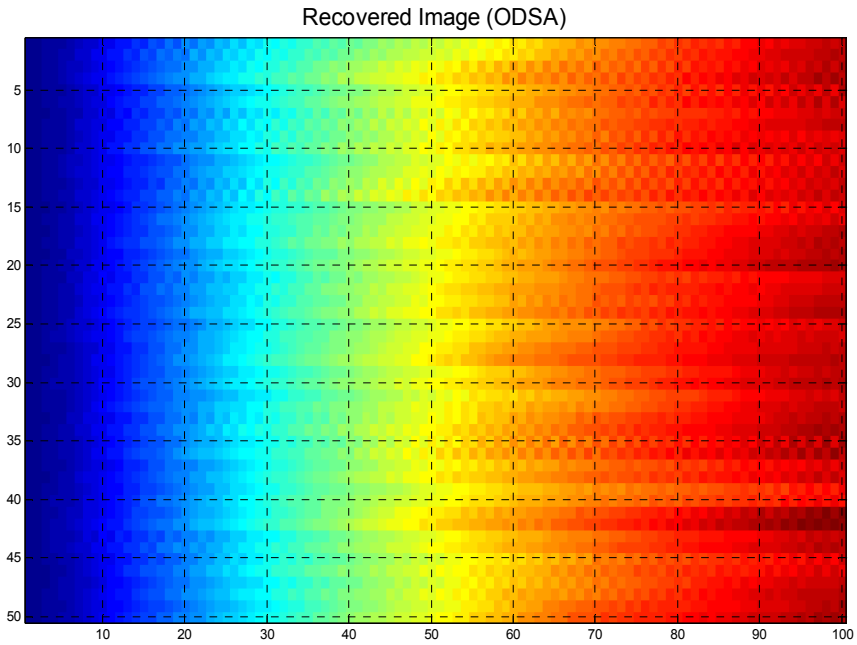

Figure 6-61 The Estimated Image by using the modified version of ODSA

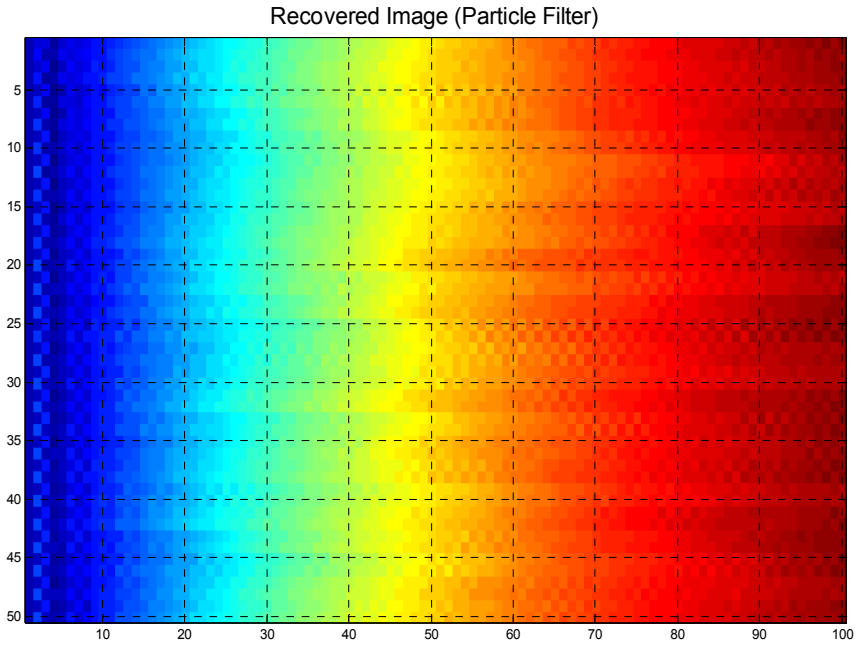

Figure 6-62 The Estimated Image by using the Bootstrap Filter Algorithm

The resulting mean_absolute_error for ODSA estimation is 8.9803 and the relative error is 11.72%. The time ODSA took for the estimation process is 14.4445 seconds per image.

When the Bootstrap Filter Algorithm was performed the mean_absolute_error became 4.4583 and the relative error was 5.90%. The processing time for the Bootstrap Filter was 15.6701 seconds per image.

6.3.3. Simulation 3

This time the parameters were set as the following:

max_x_k_size	:50
gate_size	:0.1
no_of_sampling_points_x	:3

no_of_sampling_points_w :3

Ns :150

With the parameters given above, the following results were obtained:

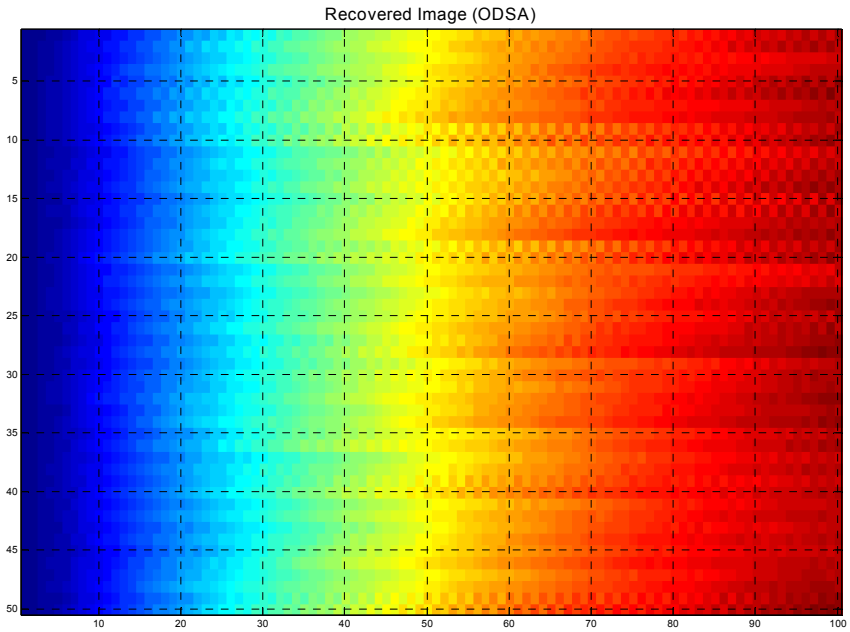

Figure 6-63 The Estimated Image by using the modified version of ODSA

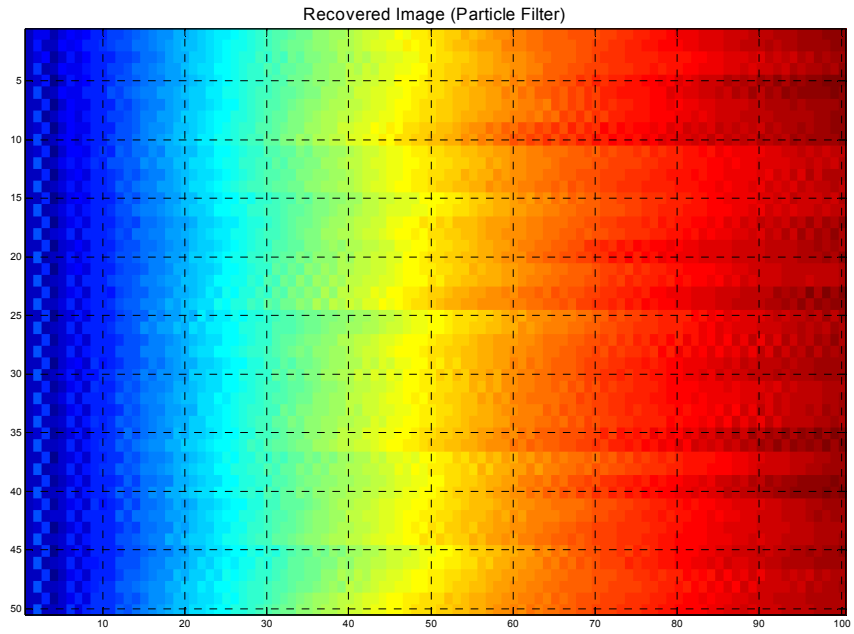

Figure 6-64 The Estimated Image by using the Bootstrap Filter Algorithm

The resulting mean_absolute_error for ODSA estimation is 9.0149 and the relative error is 11.77%. The time ODSA took for the estimation process is 33.9186 seconds per image.

With the Bootstrap Filter Algorithm the mean_absolute_error became 4.1038 and the relative error was 5.42%. The processing time for the Bootstrap Filter was 32.9741 seconds per image.

6.3.4. Simulation 4

For this simulation the parameters were changed to the following values:

max_x_k_size	:81
gate_size	:0.1
no_of_sampling_points_x	:3

no_of_sampling_points_w :3

Ns :250

The results are given below:

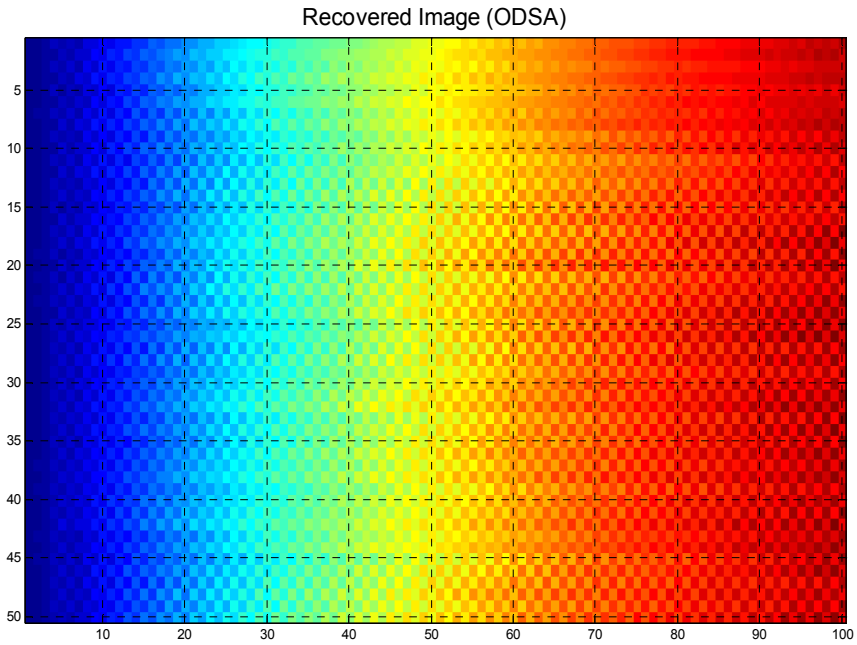

Figure 6-65 The Estimated Image by using the modified version of ODSA

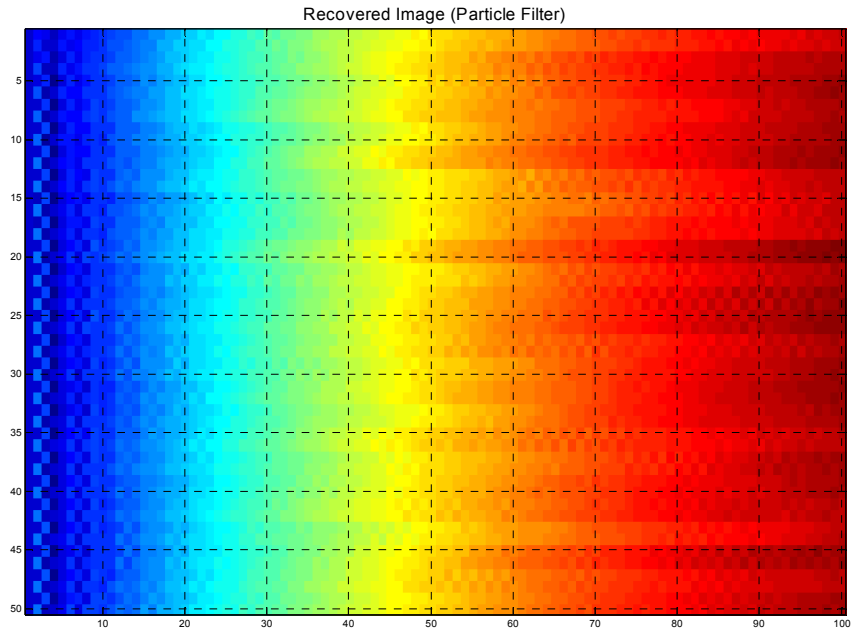

Figure 6-66 The Estimated Image by using the Bootstrap Filter Algorithm

For ODSA estimation, the resulting mean_absolute_error is 9.1216 and the relative error is 11.88%. The time ODSA took for the estimation process is 70.7059 seconds per image.

With the Bootstrap Filter Algorithm the mean_absolute_error became 3.7435 and the relative error was 4.95%. The processing time for the Bootstrap Filter was 86.9767 seconds per image.

6.3.5. Simulation 5

So far, the limit on the number of state nodes was increased to see its effect on the performance of the modified version of ODSA. This time the node limit was kept as the same but the gate size was varied. The simulation parameters for both of the algorithms are as the following:

max_x_k_size	:81
gate_size	:0.01
no_of_sampling_points_x	:3
no_of_sampling_points_w	:3
Ns	:500

The results with these parameters can be seen below:

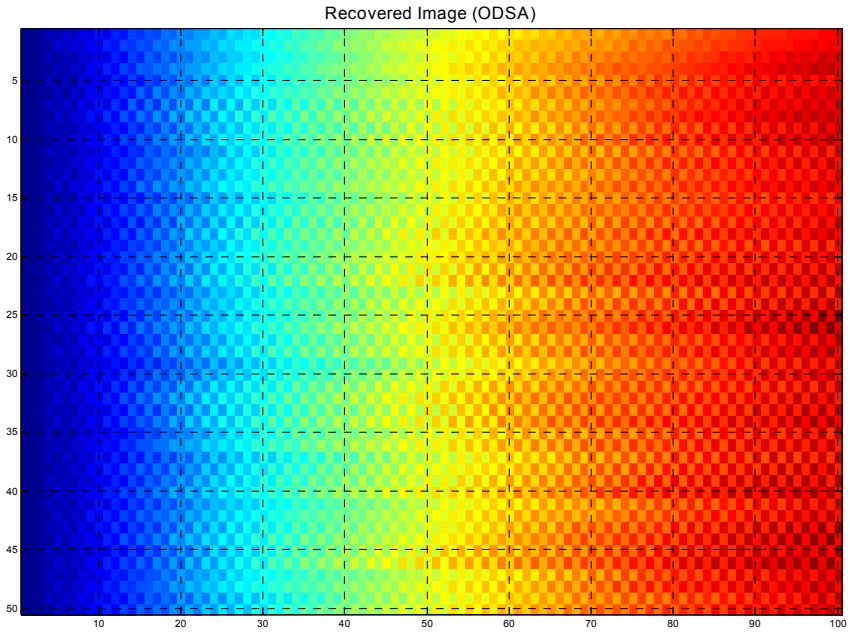

Figure 6-67 The Estimated Image by using the modified version of ODSA

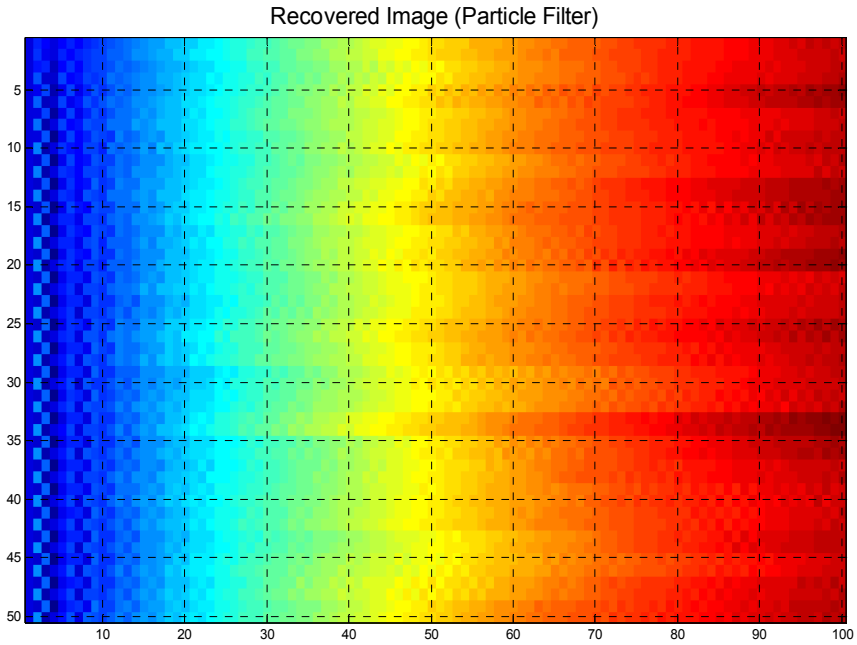

Figure 6-68 The Estimated Image by using the Bootstrap Filter Algorithm

The resulting mean_absolute_error for ODSA estimation is 9.1241 and the relative error is 11.97%. The time ODSA took for the estimation process is 61.4818 seconds per image.

When the Bootstrap Filter Algorithm was performed the mean_absolute_error became 3.3108 and the relative error was 4.40%. The processing time for the Bootstrap Filter was 337.1435 seconds per image.

6.3.6. Simulation 6

This time the following parameters were used for the image restoration process:

max_x_k_size	:81
gate_size	:0.25
no_of_sampling_points_x	:3

no_of_sampling_points_w :3

Ns :750

The results are given below:

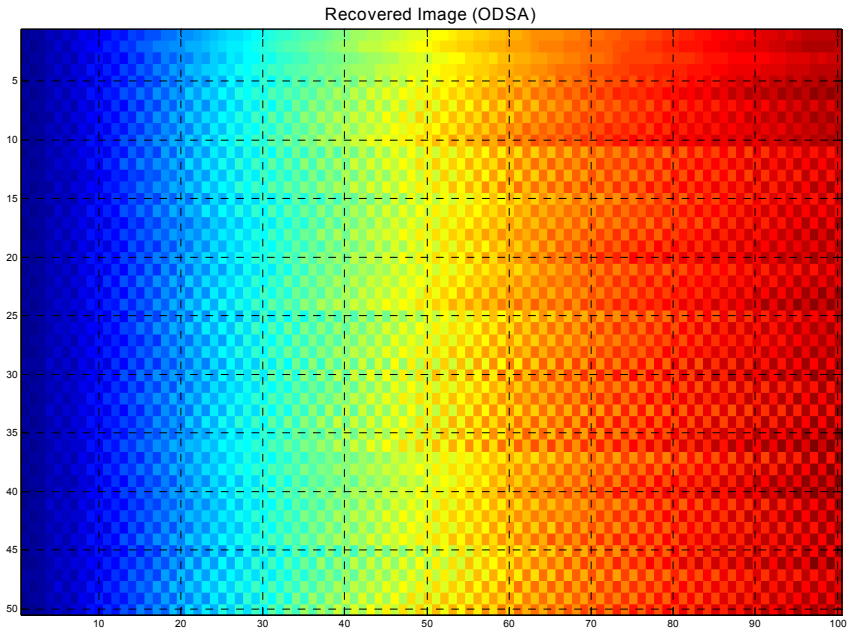

Figure 6-69 The Estimated Image by using the modified version of ODSA

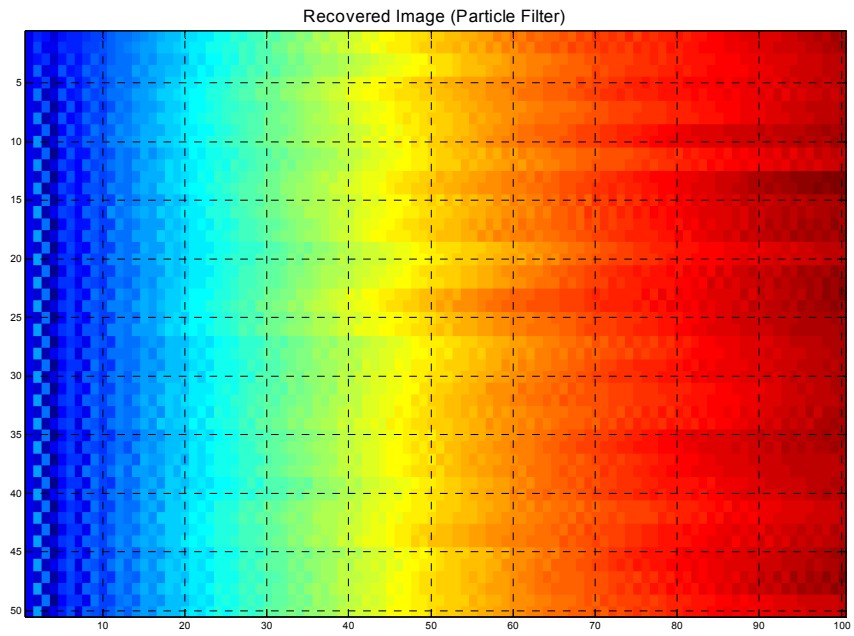

Figure 6-70 The Estimated Image by using the Bootstrap Filter Algorithm

For ODSA estimation, the resulting mean_absolute_error is 9.2351 and the relative error is 12.02%. The time ODSA took for the estimation process is 97.4324 seconds per image.

With the Bootstrap Filter Algorithm the mean_absolute_error became 3.0346 and the relative error was 4.02%. The processing time for the Bootstrap Filter was 757.2554 seconds per image.

6.3.7. Simulation 7

From this simulation on, the image restoration process was performed only by using the modified version of ODSA. The mentioned simulations were performed with different number of possible values of the initial state variable, x(0) and the disturbance noise variable, w(k). The parameters used are listed below:

max_x_k_size :50

gate_size :0.1

no_of_sampling_points_x :5

no_of_sampling_points_w :5

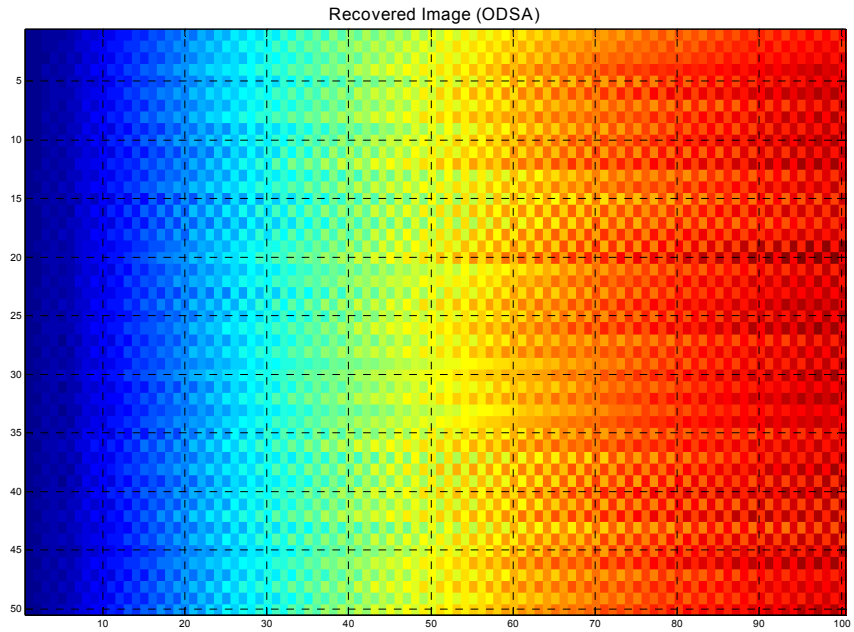

Figure 6-71 The Estimated Image by using the modified version of ODSA

The resulting mean_absolute_error is 14.5112 and the relative error is 18.82%. The time ODSA took for the estimation process is 167.9748 seconds per image in this case.

6.3.8. Simulation 8

The last simulation for this model was performed with the following parameters:

max_x_k_size :50

gate_size :0.1

no_of_sampling_points_x :7

no_of_sampling_points_w :7

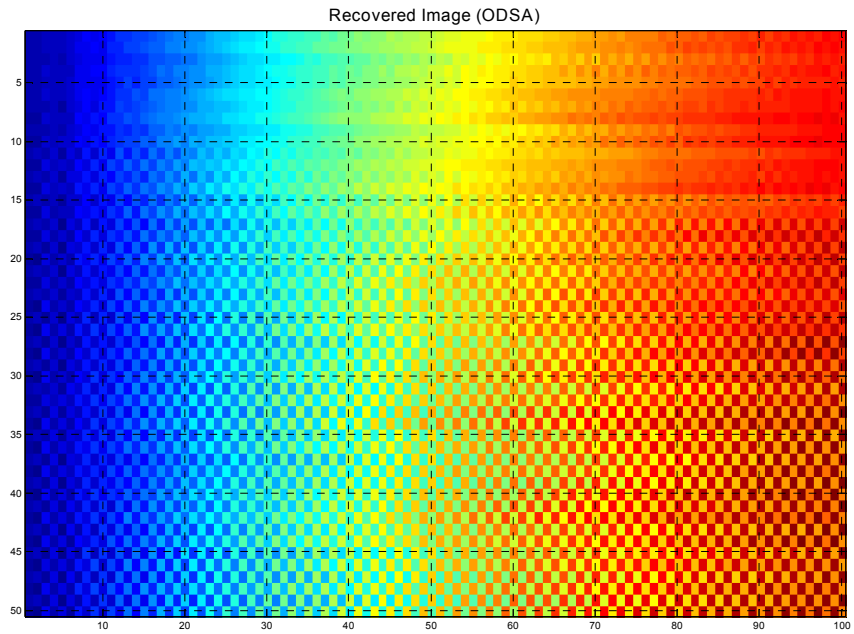

Figure 6-72 The Estimated Image by using the modified version of ODSA

The resulting mean_absolute_error for this case is 21.8331 and the relative error is 29.03%. The time ODSA took for the estimation process is 513.5193 seconds per image.

6.3.9. Summary of the Results

The graphics showing the error and process time performance of the estimation algorithms with respect to the change in the parameter values are given below:

Timing Performance Graphics:

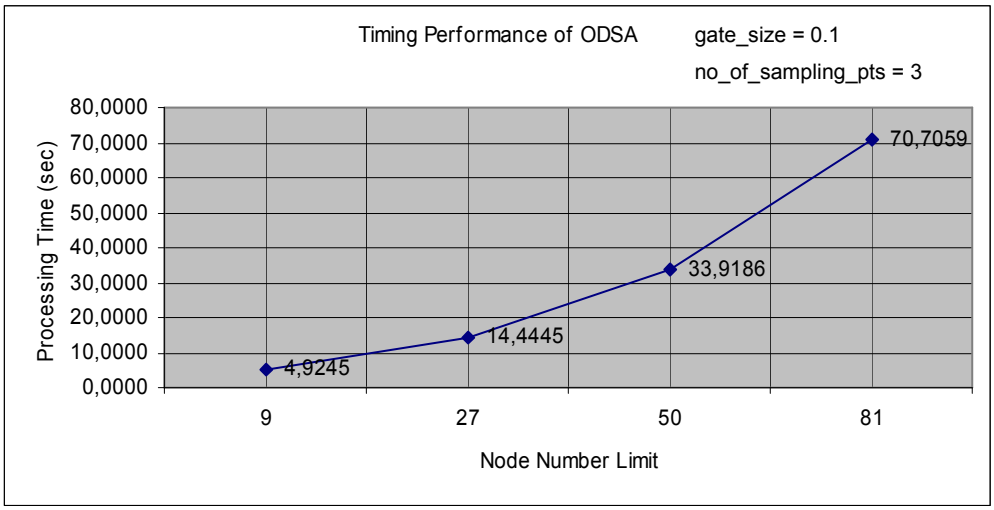

Figure 6-73 Timing Performance of ODSA with the parameters given on the graph

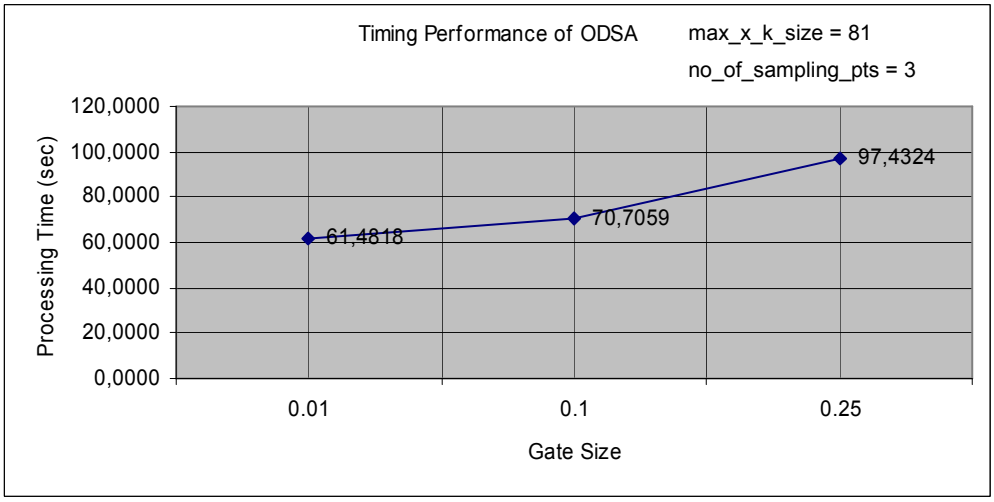

Figure 6-74 Timing Performance of ODSA with the parameters given on the graph

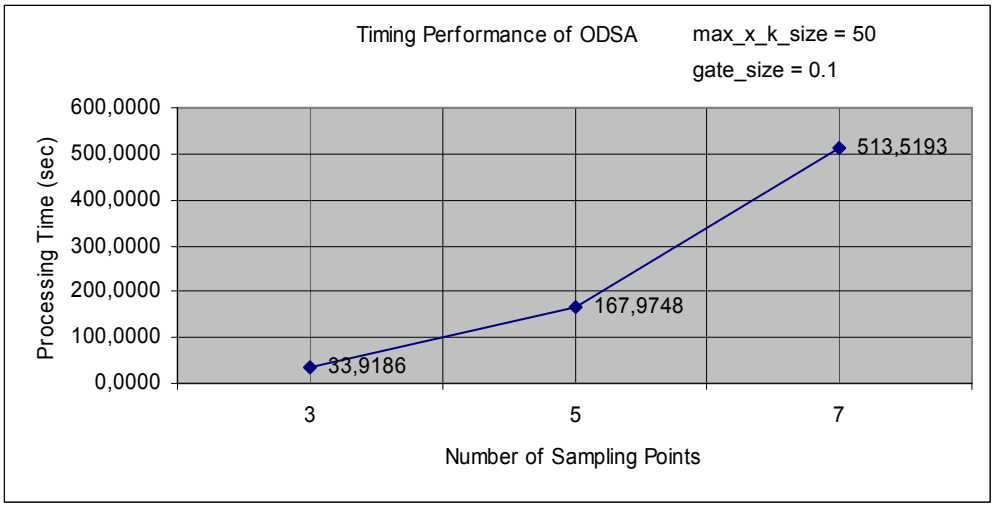

Figure 6-75 Timing Performance of ODSA with the parameters given on the graph

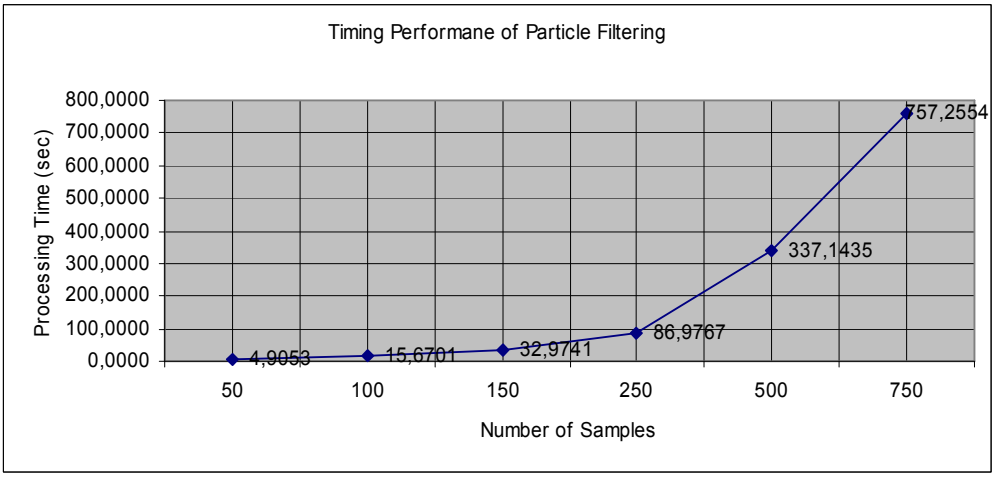

Figure 6-76 Timing Performance of the Bootstrap Filter Algorithm w.r.t. Ns

Error Performance Graphics:

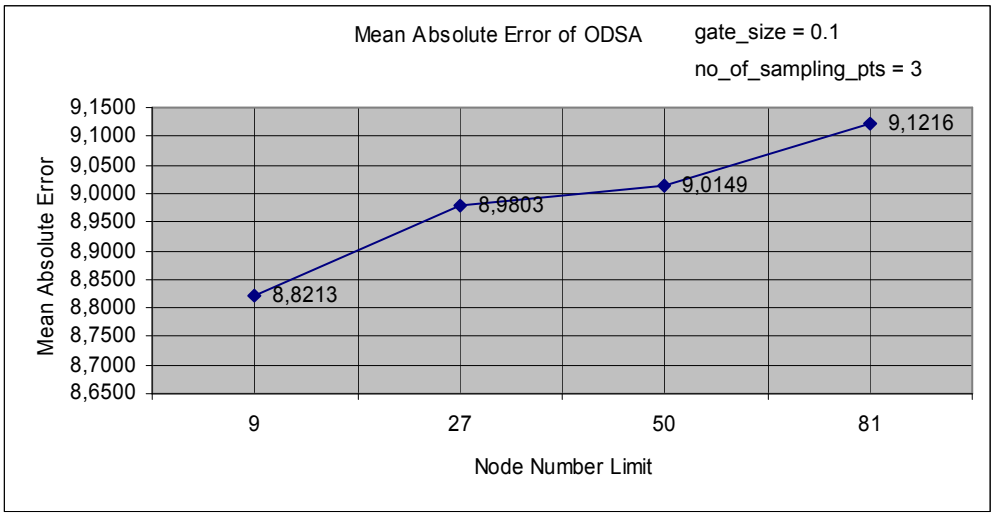

Figure 6-77 Mean absolute error per pixel for ODSA w.r.t. the parameters given on the graph

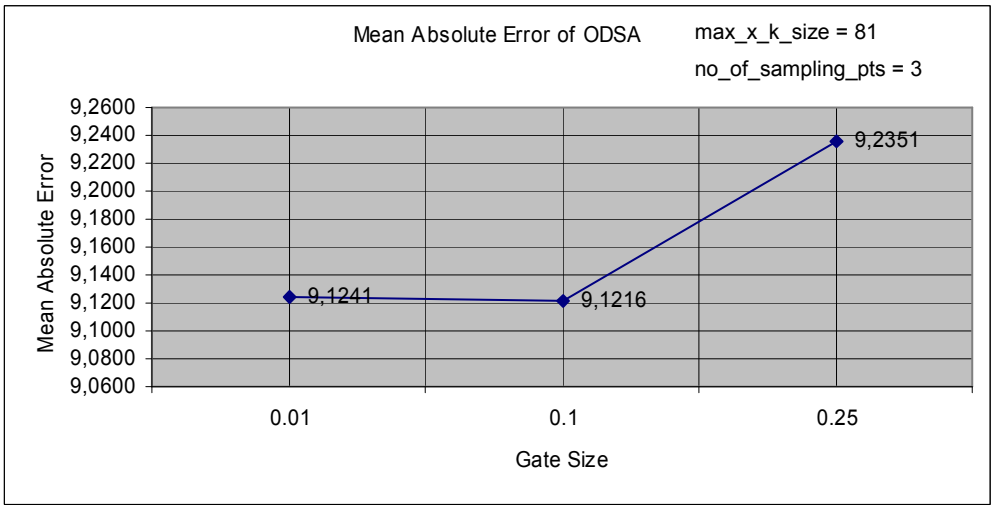

Figure 6-78 Mean absolute error per pixel for ODSA w.r.t. the parameters given on the graph

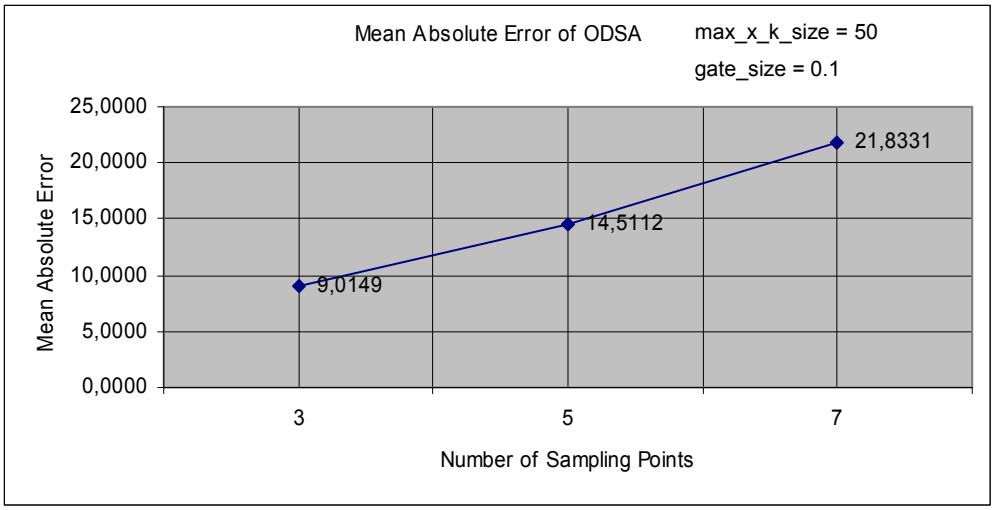

Figure 6-79 Mean absolute error per pixel for ODSA w.r.t. the parameters given on the graph

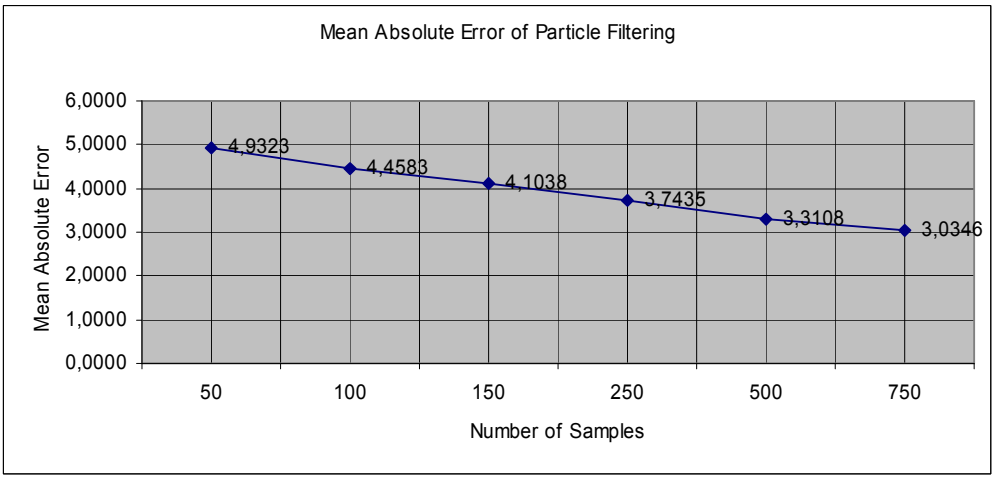

Figure 6-80 Mean absolute error per pixel for the Bootstrap Filter w.r.t Ns

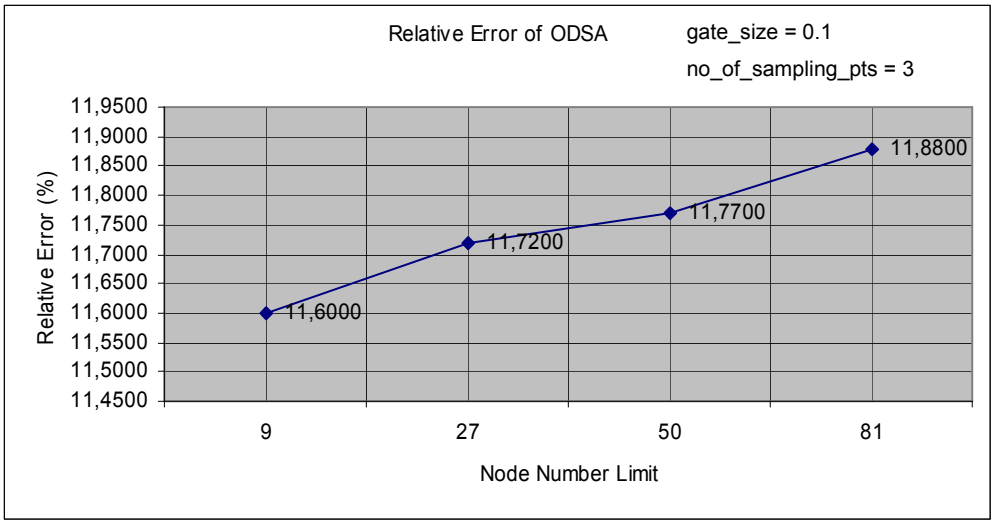

Figure 6-81 Relative error per pixel for ODSA w.r.t. the parameters given on the graph

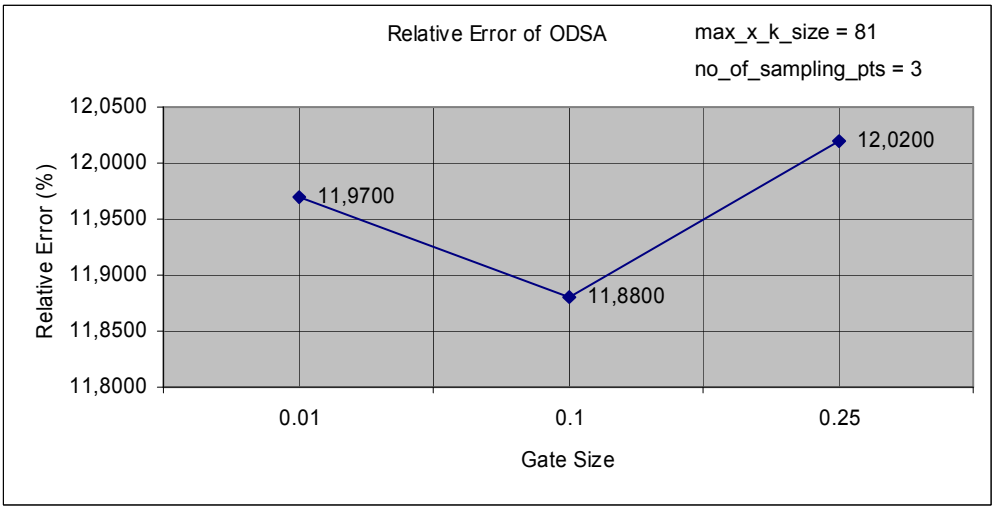

Figure 6-82 Relative error per pixel for ODSA w.r.t. the parameters given on the graph

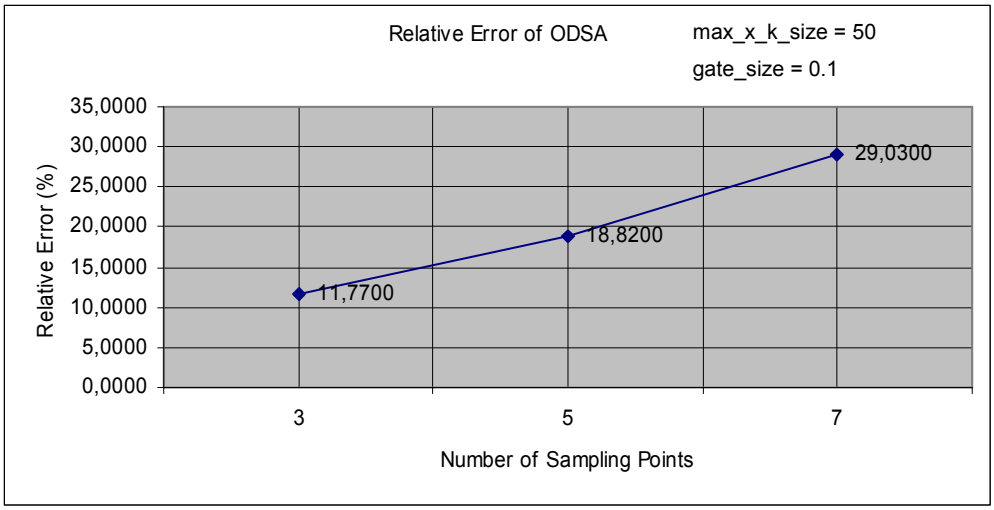

Figure 6-83 Relative error per pixel for ODSA w.r.t. the parameters given on the graph

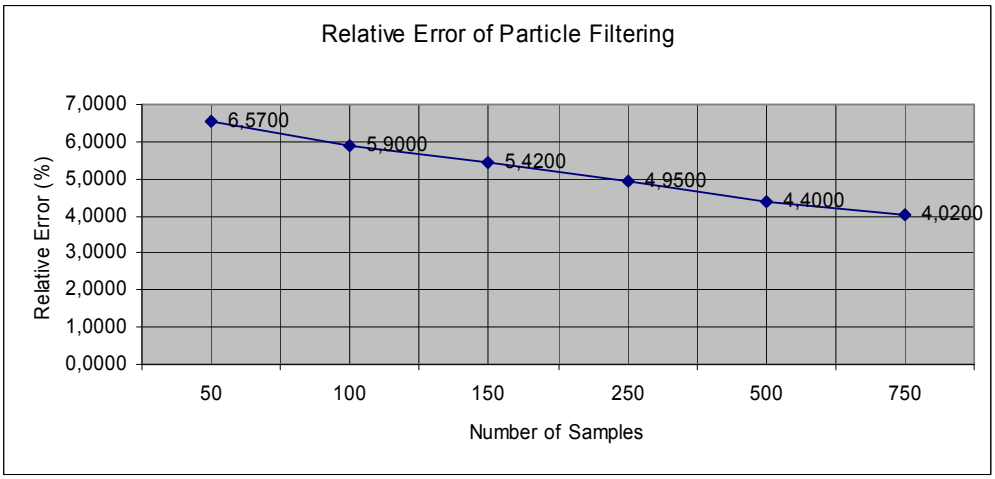

Figure 6-84 Mean absolute error per pixel for the Bootstrap Filter w.r.t Ns

The original image was generated randomly for each of the simulations. Therefore, relative_error value rather than mean_absolute_error should be used for the comparison of error performance of different simulations.

By using ODSA the best estimation for the original image has 11.60% relative error per pixel. This result was found in 4.9245 seconds for a single image on average. Also this one has the best timing performance, i.e. it has

the lowest processing time. The best estimation obtained with the Bootstrap Filter has 4.02% relative error per pixel. This result was obtained in 757.2554 seconds for a single image on average. The fastest estimation with the Bootstrap Filter Algorithm has been reached in 4.9053 seconds for a single image on average with 6.57% relative error per pixel.

6.4. Model 4

This time the image to be processed has the following state transition equations:

$$G_1(x_1(k), x_2(k), w_1(k)) = |x_1(k) - x_2(k)|^{\frac{2}{3}} + w_1^{\,2}(k), \tag{6.18}$$

$$G_2(x_1(k), x_2(k), w_2(k)) = x_1(k)\exp\left(-0.1\left[\frac{x_1(k)}{|x_2(k)|+5}\right]\right) + 20 + w_2(k). \tag{6.19}$$

and the point spread function of the pixels assumed to be:

$$h(x, y, m, n, f(m,n)) = \left[(x^2 + y^2)mnf(m,n)\right]^{\frac{1}{3}}. \tag{6.20}$$

For this model the mean of the initial state, x(0) was assumed to be 50 and the variance of it was chosen as 3. The process noise, w(k) and the observation noise v(k) were taken as zero mean Gaussian random variables with variances 2.5 and 1.25 respectively. All the simulations for this model were performed by using MATLAB version 7 running on a computer with an Intel Xeon – 3.4GHz processor and 4GB of RAM. For each simulation the estimation process for an image was performed for 250 times and the average of the results were calculated.

The original image and the blurred version of the original image according to the given models and statistics given above are presented in the figures 6-85 and 6-86 respectively.

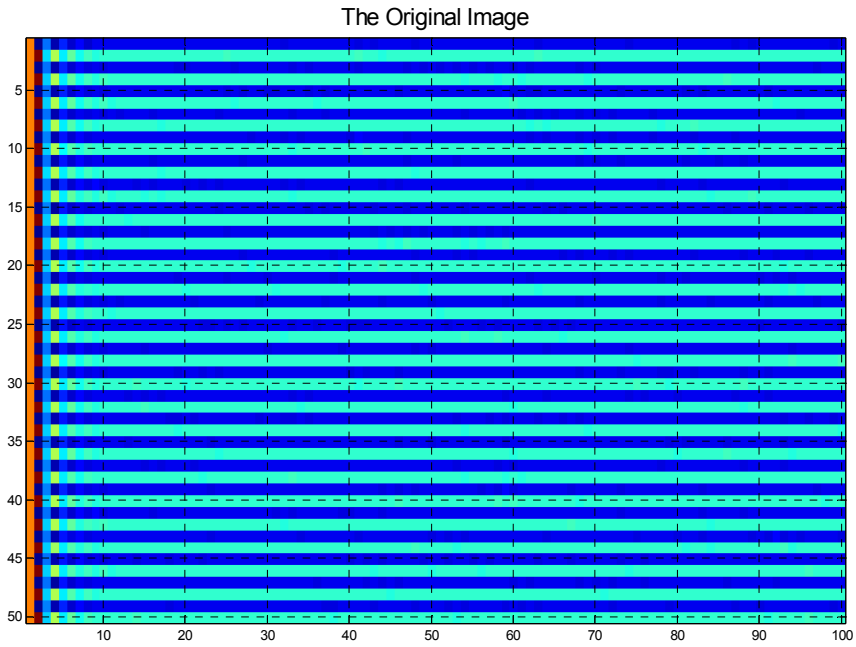

Figure 6-85 The Original Image according to the state transition equation

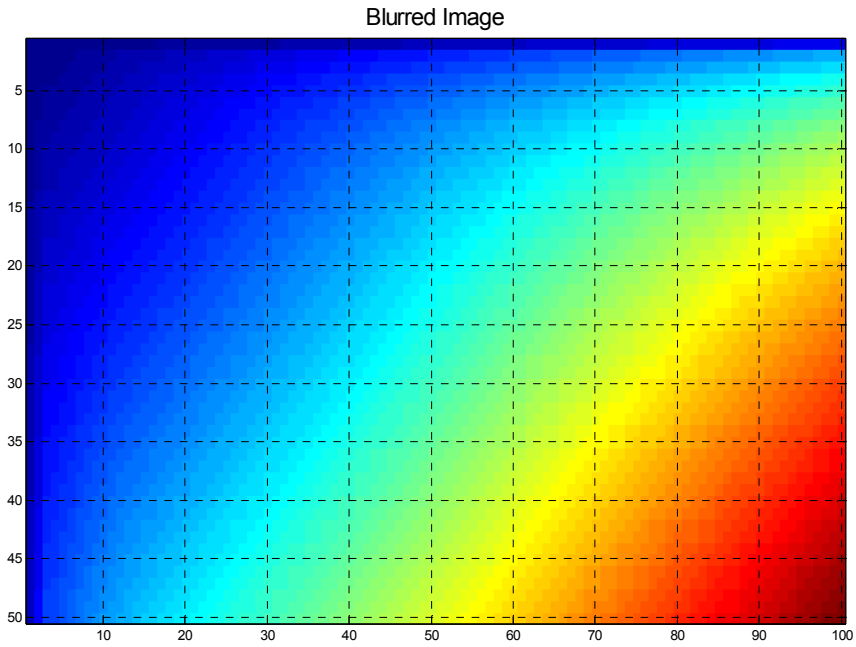

Figure 6-86 Result of the blurring function

6.4.1. Simulation 1

The results of the estimation algorithms with the parameters given below are as the following:

max_x_k_size	:9
gate_size	:0.1
no_of_sampling_points_x	:3
no_of_sampling_points_w	:3
Ns	:50

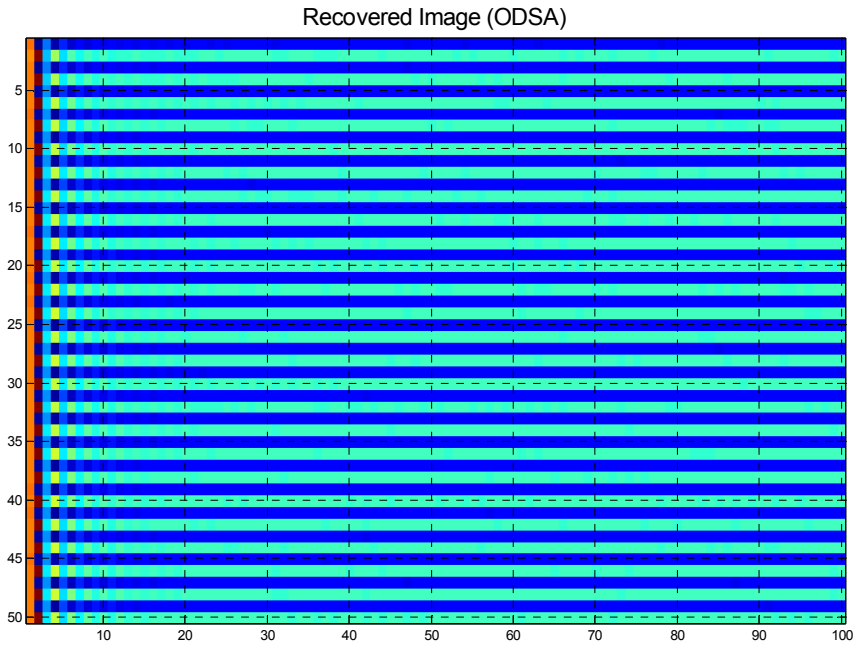

Figure 6-87 The Estimated Image by using the modified version of ODSA

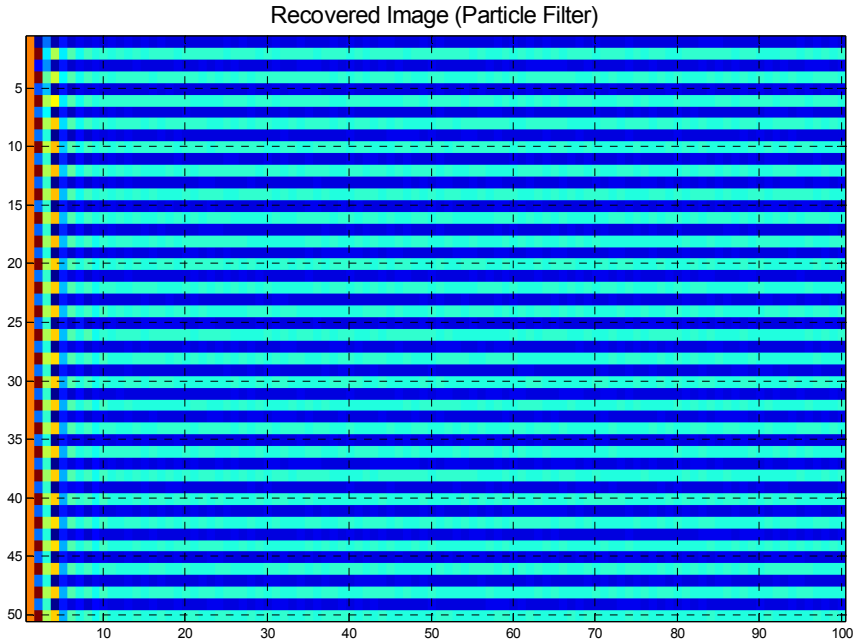

Figure 6-88 The Estimated Image by using the Bootstrap Filter Algorithm

The resulting mean_absolute_error for ODSA estimation is 2.6330 and the relative error is 13.14%. The time ODSA took for the estimation process is 6.6387 seconds per image.

When the Bootstrap Filter Algorithm was performed the mean_absolute_error became 1.8106 and the relative error was 9.04%. The processing time for the Bootstrap Filter was 3.4105 seconds per image.

6.4.2. Simulation 2

The second simulation for this model was performed with the following parameters:

max_x_k_size :27
gate_size :0.1
no_of_sampling_points_x :3

no_of_sampling_points_w :3

Ns :100

The estimated images and the performance of the two algorithms are given below:

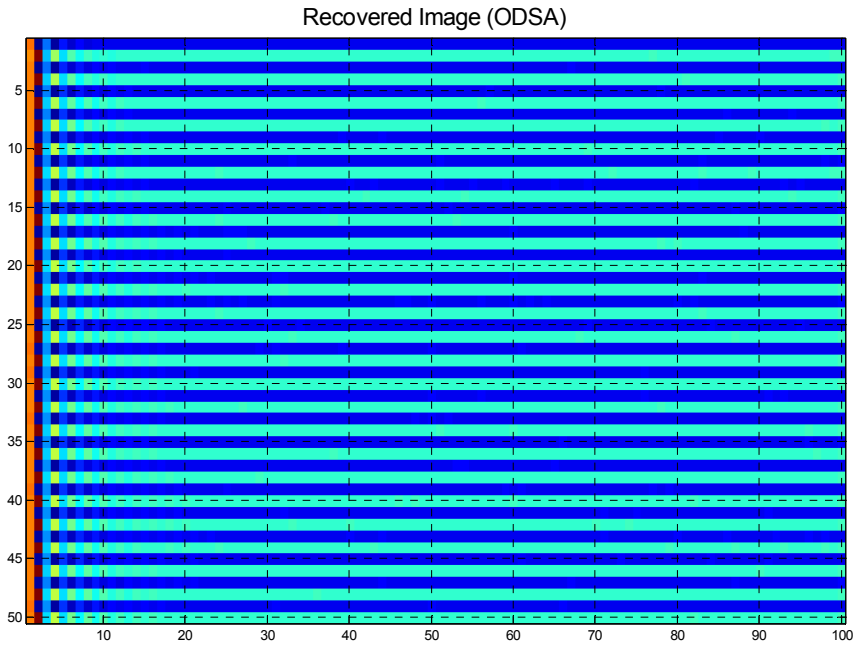

Figure 6-89 The Estimated Image by using the modified version of ODSA

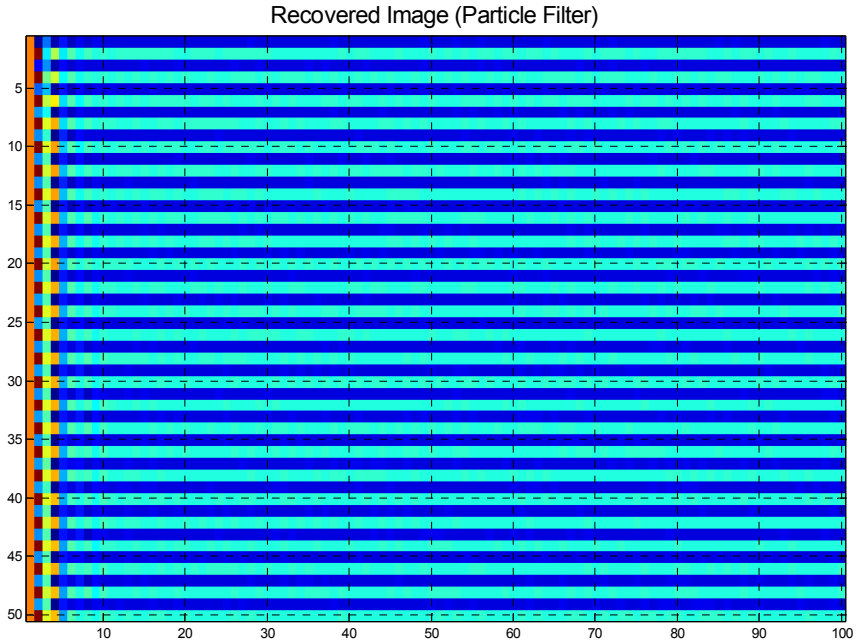

Figure 6-90 The Estimated Image by using the Bootstrap Filter Algorithm

The resulting mean_absolute_error for ODSA estimation is 2.6442 and the relative error is 13.20%. The time ODSA took for the estimation process is 21.1782 seconds per image.

When the Bootstrap Filter Algorithm was performed the mean_absolute_error became 1.7574 and the relative error was 8.77%. The processing time for the Bootstrap Filter was 8.4065 seconds per image.

6.4.3. Simulation 3

This time the parameters were set as the following:

max_x_k_size	:50
gate_size	:0.1
no_of_sampling_points_x	:3

no_of_sampling_points_w :3

Ns :150

With the parameters given above, the following results were obtained:

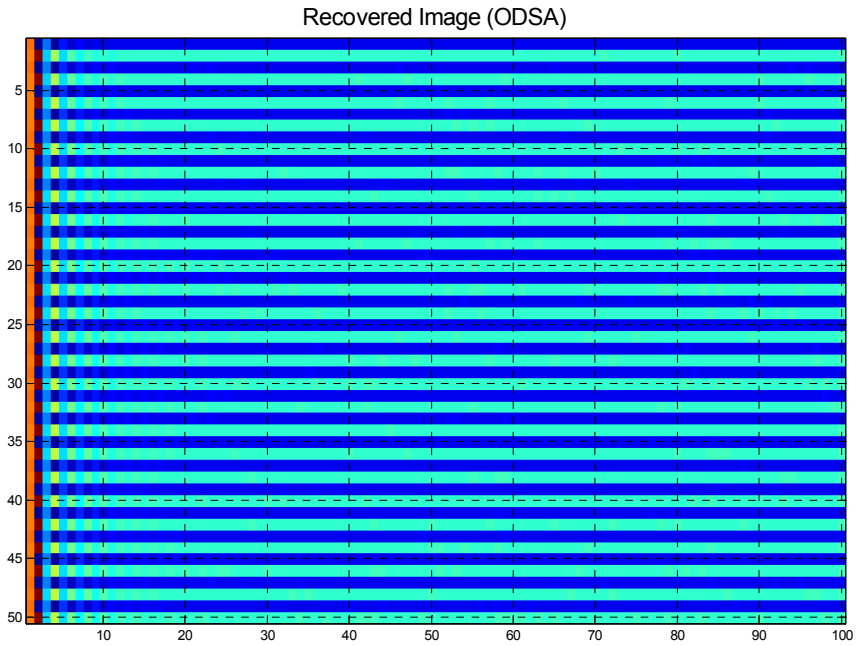

Figure 6-91 The Estimated Image by using the modified version of ODSA

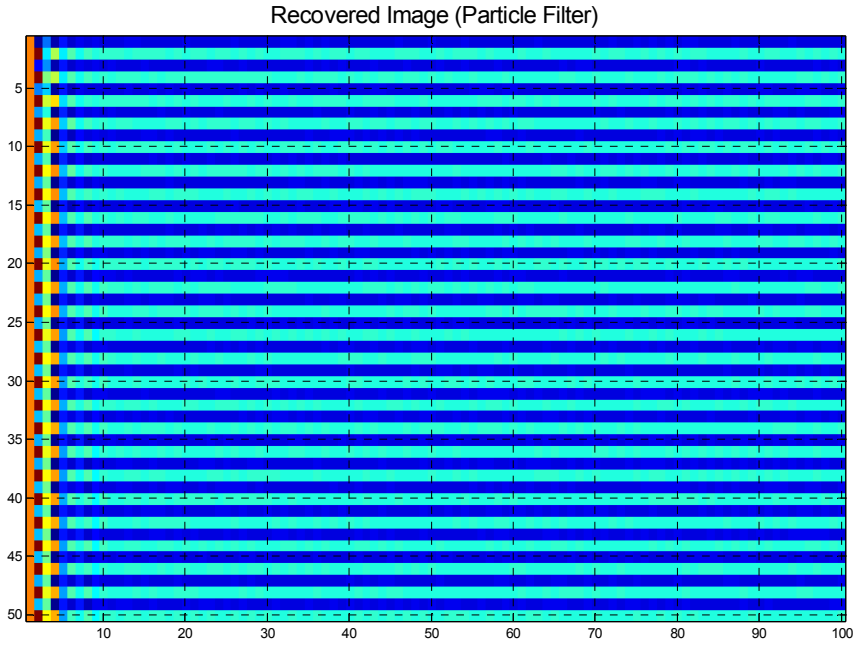

Figure 6-92 The Estimated Image by using the Bootstrap Filter Algorithm

The resulting mean_absolute_error for ODSA estimation is 2.6496 and the relative error is 13.22%. The time ODSA took for the estimation process is 50.3531 seconds per image.

With the Bootstrap Filter Algorithm the mean_absolute_error became 1.7380 and the relative error was 8.67%. The processing time for the Bootstrap Filter was 15.2166 seconds per image.

6.4.4. Simulation 4

For this simulation the parameters were changed to the following values:

max_x_k_size	:81
gate_size	:0.1
no_of_sampling_points_x	:3

no_of_sampling_points_w :3

Ns :250

The results are given below:

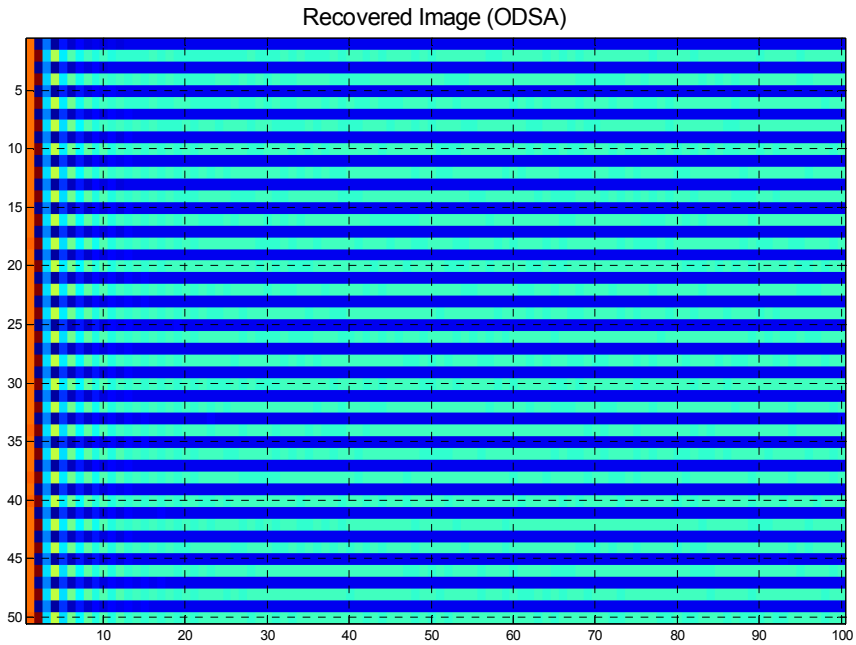

Figure 6-93 The Estimated Image by using the modified version of ODSA

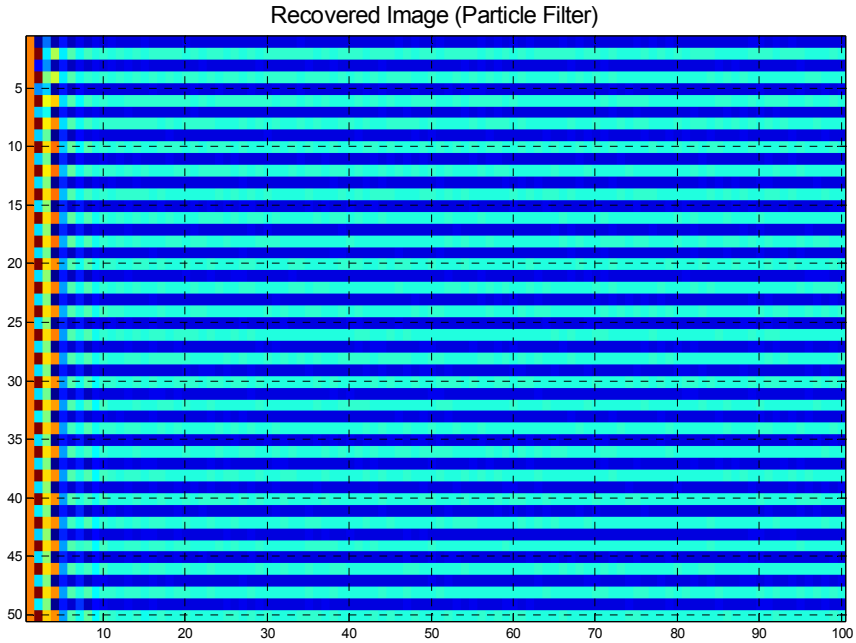

Figure 6-94 The Estimated Image by using the Bootstrap Filter Algorithm

For ODSA estimation, the resulting mean_absolute_error is 2.6468 and the relative error is 13.21%. The time ODSA took for the estimation process is 106.6170 seconds per image.

With the Bootstrap Filter Algorithm the mean_absolute_error became 1.7532 and the relative error was 8.74%. The processing time for the Bootstrap Filter was 35.5606 seconds per image.

6.4.5. Simulation 5

So far, the limit on the number of state nodes was increased to see its effect on the performance of the modified version of ODSA. This time the node limit was kept as the same but the gate size was varied. The simulation parameters for both of the algorithms are as the following:

max_x_k_size	:81
gate_size	:0.01
no_of_sampling_points_x	:3
no_of_sampling_points_w	:3
Ns	:500

The results with these parameters can be seen below:

Figure 6-95 The Estimated Image by using the modified version of ODSA

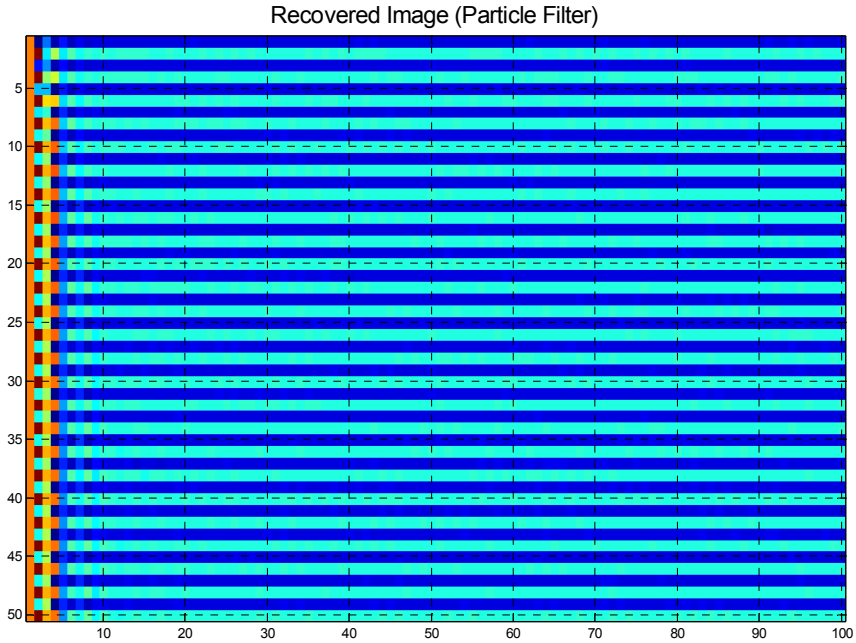

Figure 6-96 The Estimated Image by using the Bootstrap Filter Algorithm

The resulting mean_absolute_error for ODSA estimation is 2.6473 and the relative error is 13.20%. The time ODSA took for the estimation process is 88.5887 seconds per image.

When the Bootstrap Filter Algorithm was performed the mean_absolute_error became 1.7735 and the relative error was 8.85%. The processing time for the Bootstrap Filter was 128.6142 seconds per image.

6.4.6. Simulation 6

This time the following parameters were used for the image restoration process:

max_x_k_size	:81
gate_size	:0.25
no_of_sampling_points_x	:3

no_of_sampling_points_w :3

Ns :750

The results are given below:

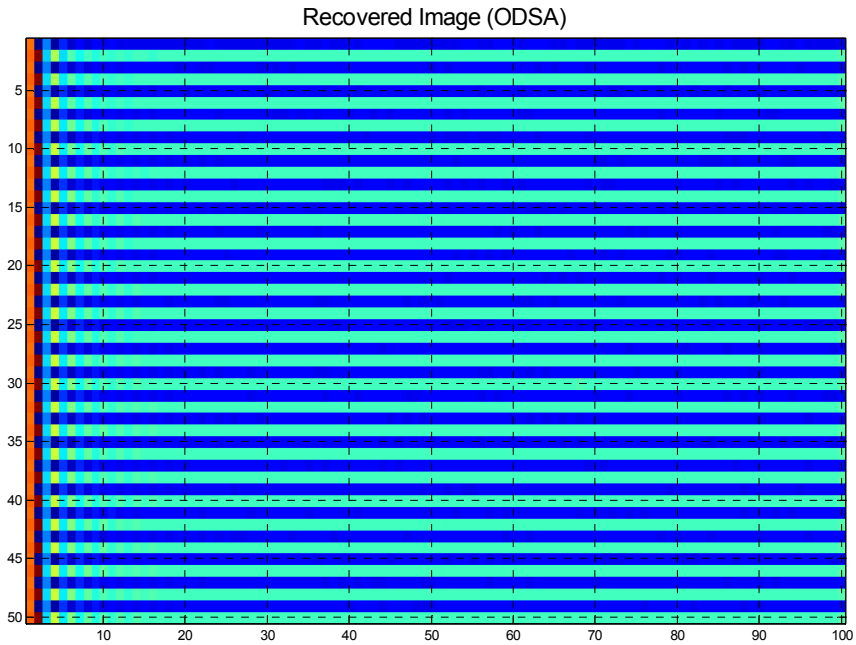

Figure 6-97 The Estimated Image by using the modified version of ODSA

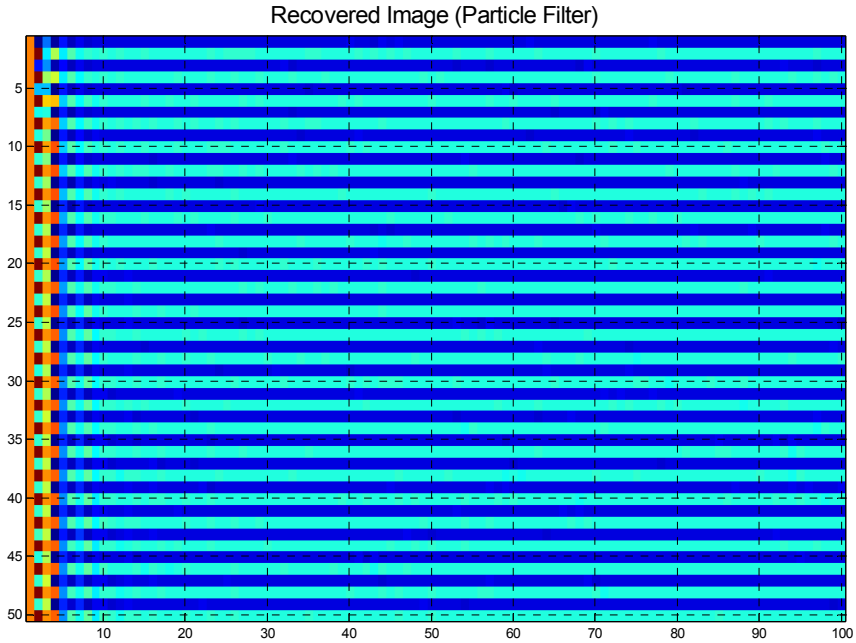

Figure 6-98 The Estimated Image by using the Bootstrap Filter Algorithm

For ODSA estimation, the resulting mean_absolute_error is 2.6060 and the relative error is 13.00%. The time ODSA took for the estimation process is 145.4024 seconds per image.

With the Bootstrap Filter Algorithm the mean_absolute_error became 1.8015 and the relative error was 8.99%. The processing time for the Bootstrap Filter was 267.8445 seconds per image.

6.4.7. Simulation 7

From this simulation on, the image restoration process was performed only by using the modified version of ODSA. The mentioned simulations were performed with different number of possible values of the initial state variable, x(0) and the disturbance noise variable, w(k). The parameters used are listed below:

max_x_k_size :50

gate_size :0.1

no_of_sampling_points_x :5

no_of_sampling_points_w :5

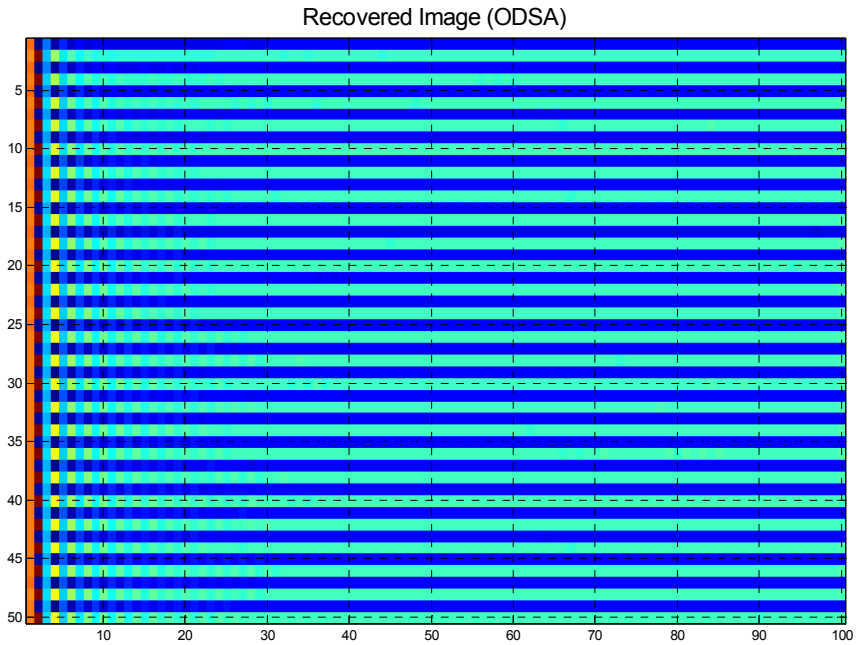

Figure 6-99 The Estimated Image by using the modified version of ODSA

The resulting mean_absolute_error is 3.1085 and the relative error is 15.51%. The time ODSA took for the estimation process is 292.5392 seconds per image in this case.

6.4.8. Simulation 8

The last simulation for this model was performed with the following parameters:

max_x_k_size :50

gate_size :0.1

no_of_sampling_points_x :7

no_of_sampling_points_w :7

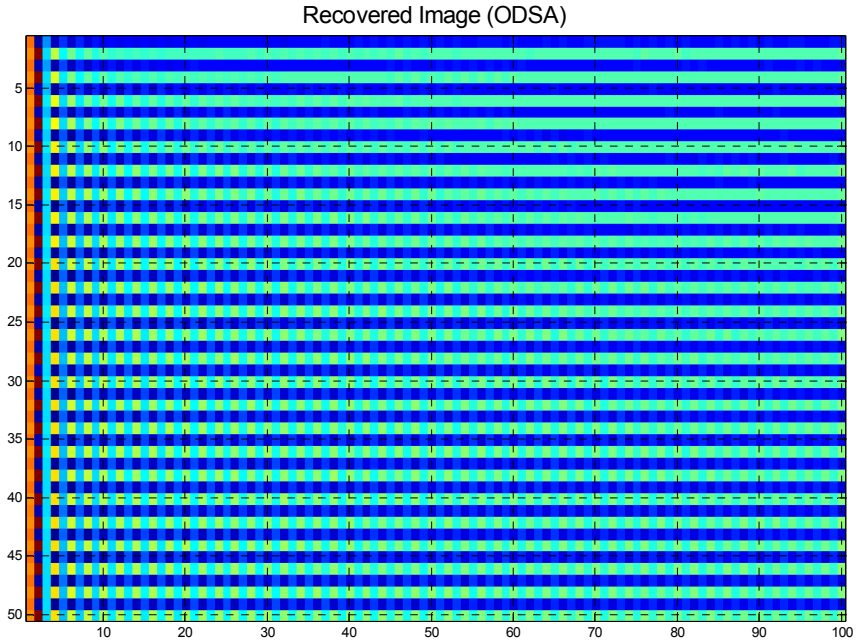

Figure 6-100 The Estimated Image by using the modified version of ODSA

The resulting mean_absolute_error for this case is 4.6446 and the relative error is 23.18%. The time ODSA took for the estimation process is 975.5654 seconds per image.

6.4.9. Summary of the Results

The graphics showing the error and process time performance of the estimation algorithms with respect to the change in the parameter values are given below:

Timing Performance Graphics:

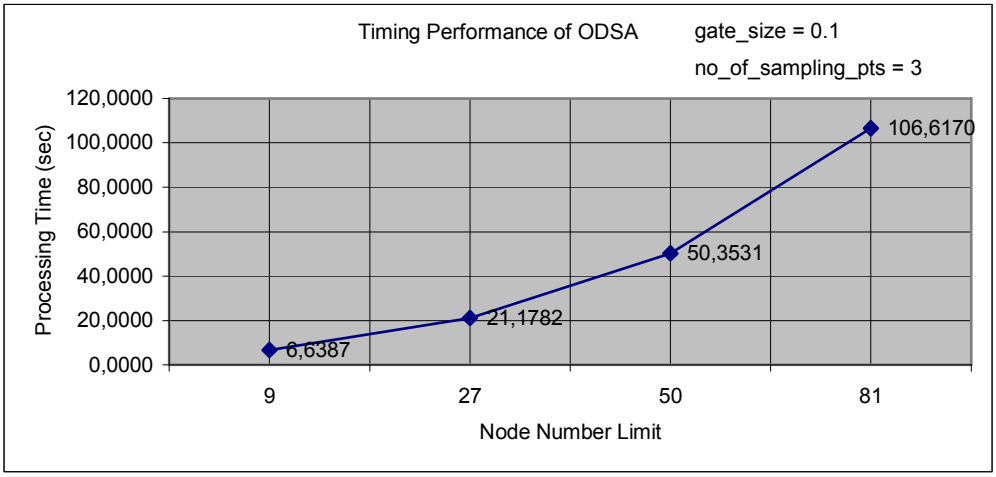

Figure 6-101 Timing Performance of ODSA with the parameters given on the graph

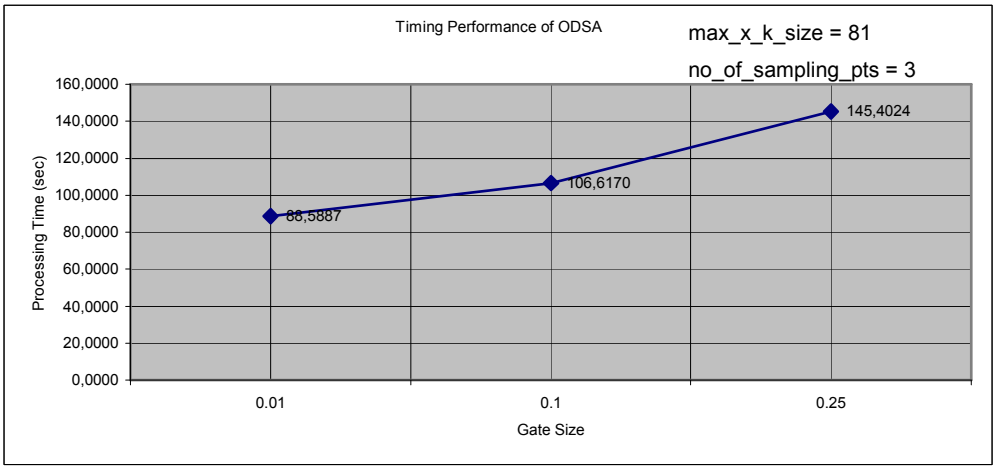

Figure 6-102 Timing Performance of ODSA with the parameters given on the graph

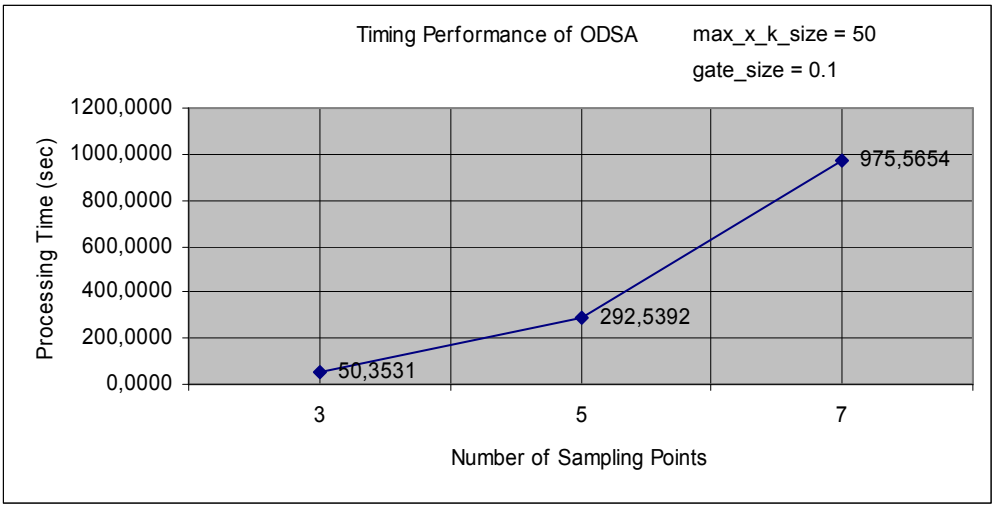

Figure 6-103 Timing Performance of ODSA with the parameters given on the graph

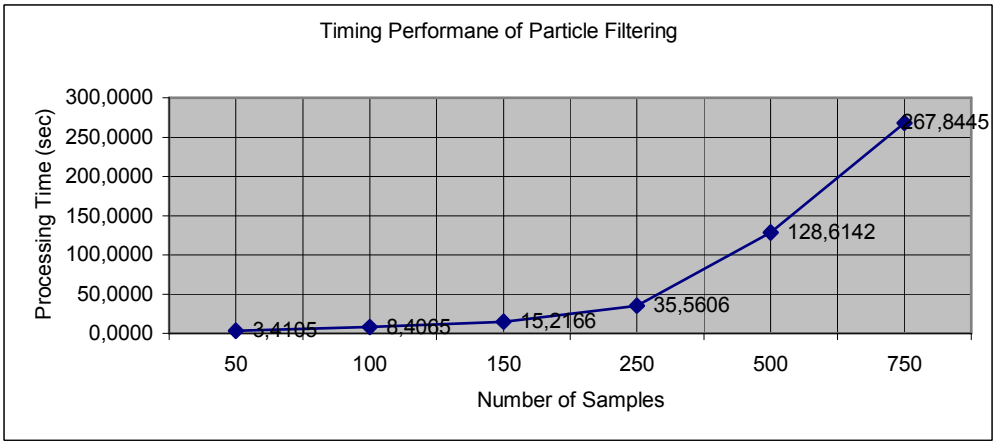

Figure 6-104 Timing Performance of the Bootstrap Filter Algorithm w.r.t. Ns

Error Performance Graphics:

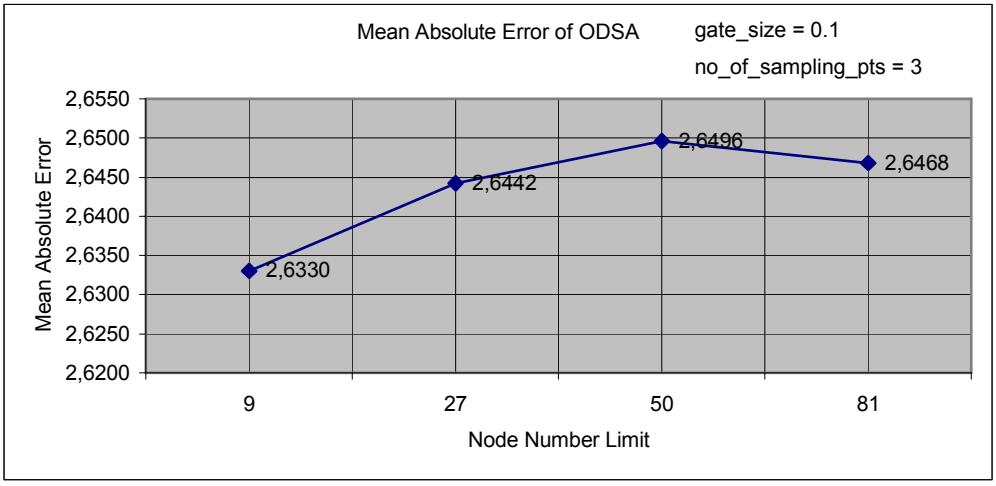

Figure 6-105 Mean absolute error per pixel for ODSA w.r.t. the parameters given on the graph

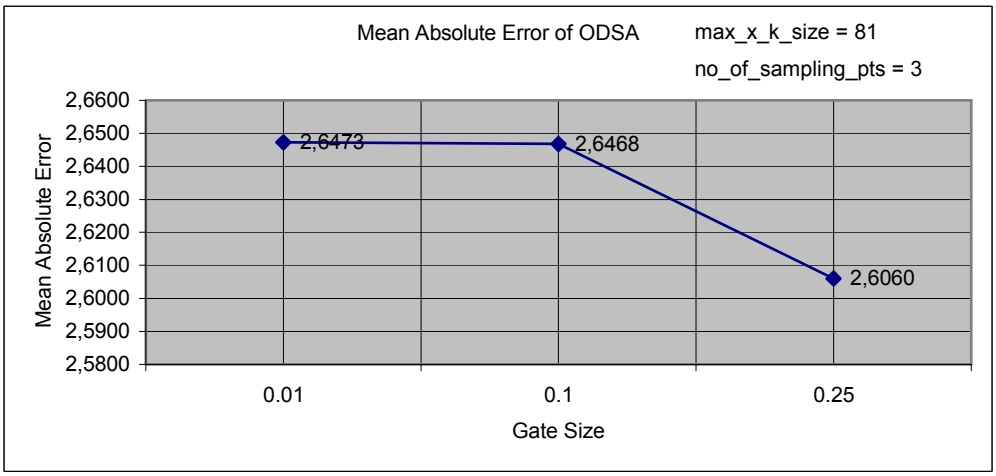

Figure 6-106 Mean absolute error per pixel for ODSA w.r.t. the parameters given on the graph

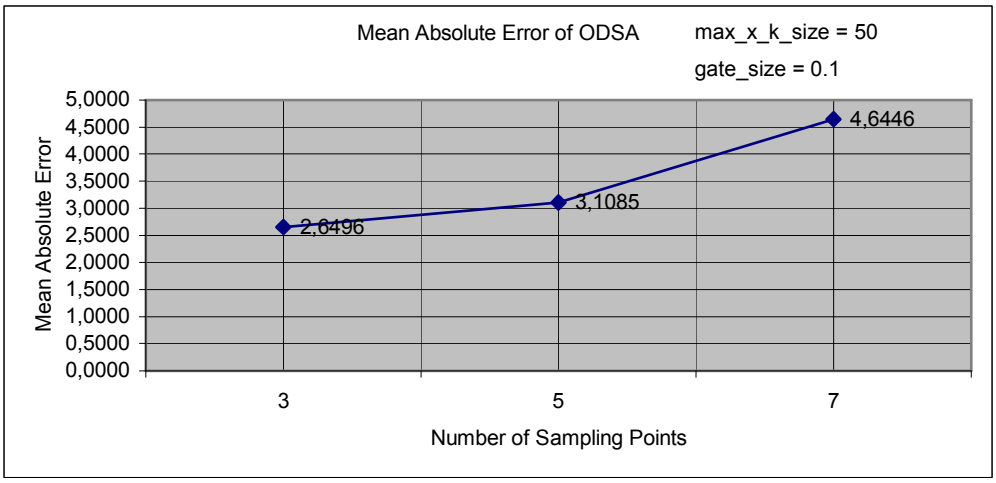

Figure 6-107 Mean absolute error per pixel for ODSA w.r.t. the parameters given on the graph

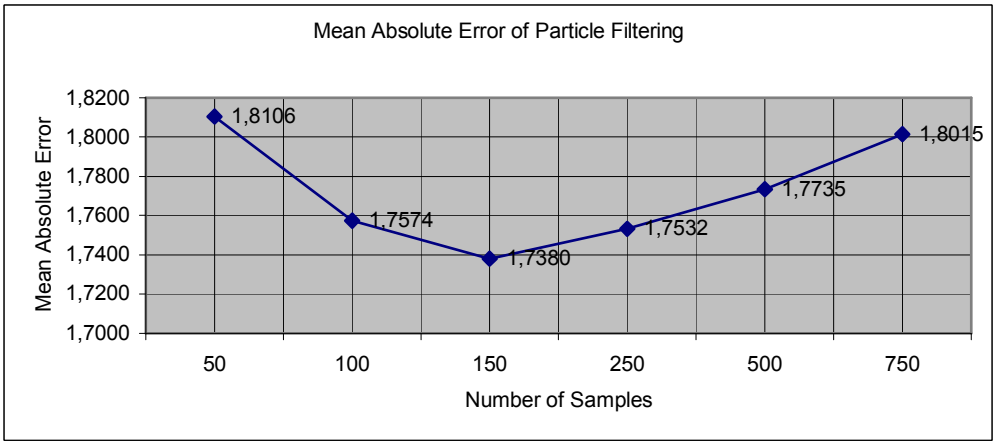

Figure 6-108 Mean absolute error per pixel for the Bootstrap Filter w.r.t Ns

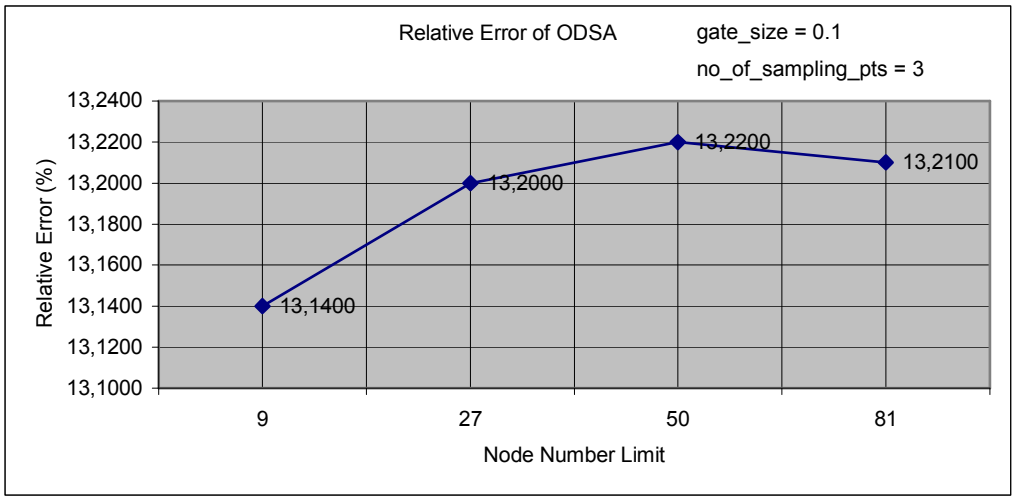

Figure 6-109 Relative error per pixel for ODSA w.r.t. the parameters given on the graph

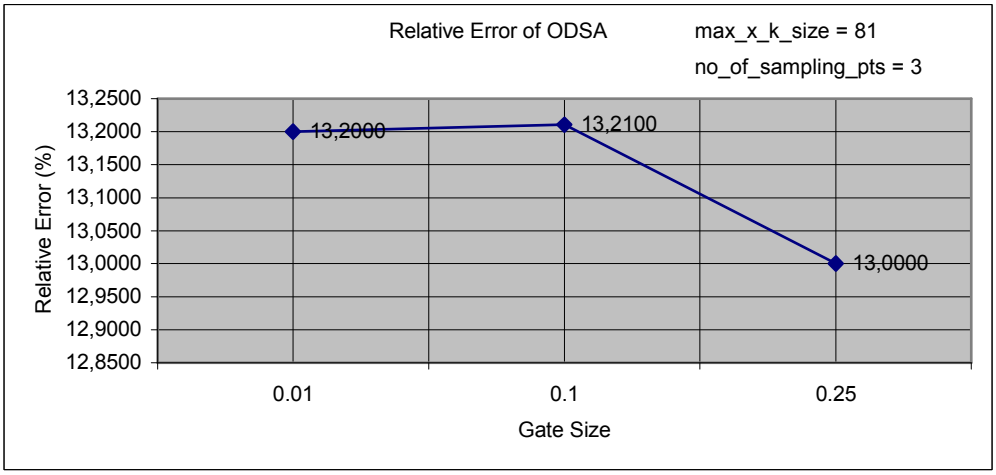

Figure 6-110 Relative error per pixel for ODSA w.r.t. the parameters given on the graph

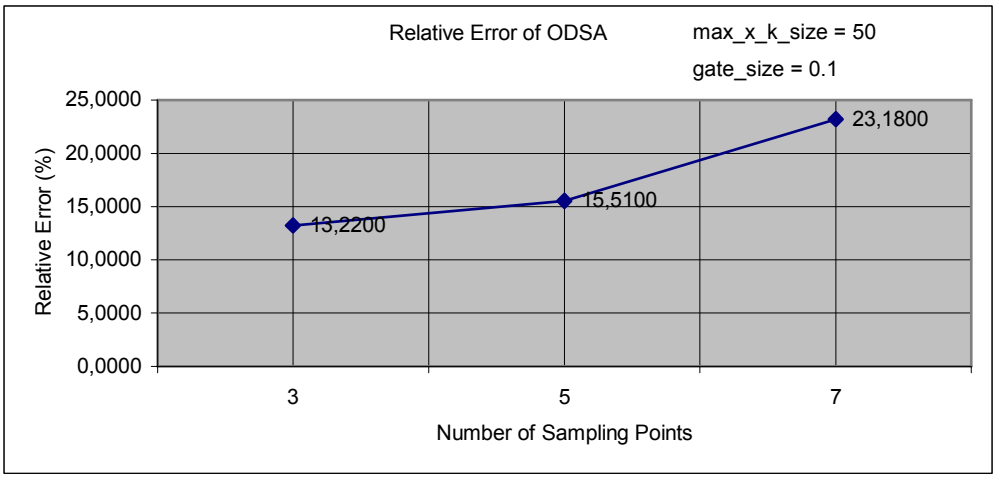

Figure 6-111 Relative error per pixel for ODSA w.r.t. the parameters given on the graph

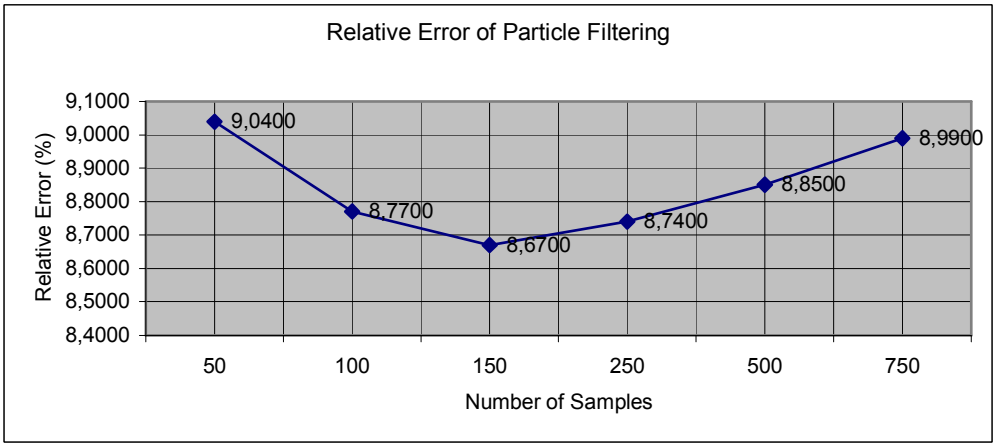

Figure 6-112 Mean absolute error per pixel for the Bootstrap Filter w.r.t Ns

The original image was generated randomly for each of the simulations. Therefore, relative_error value rather than mean_absolute_error should be used for the comparison of error performance of different simulations.

By using ODSA the best estimation for the original image has 13.00% relative error per pixel. This result was found in 145.4024 seconds for a single image on average. The fastest estimation by using ODSA gave 13.14% relative error per pixel in 6.6387 seconds for a single image on

average. The best estimation obtained with the Bootstrap Filter has 8.67% relative error per pixel. This result was obtained in 15.2166 seconds for a single image on average. The fastest estimation with the Bootstrap Filter Algorithm has been reached in 3.4105 seconds for a single image on average with 9.04% relative error per pixel.

6.5. Model 5

This time the image to be processed has the following state transition equations:

$$G_1(x_1(k), x_2(k), w_1(k)) = \frac{50|x_1(k)x_2(k)|}{x_1^2(k) + x_2^2(k) + \sqrt{|w_1(k)|}} \quad , \tag{6.21}$$

$$G_2(x_1(k), x_2(k), w_2(k)) = \left[\ln(x_1(k) + 0.5x_2(k))^4 + |x_1(k)x_2(k)|\right]^{\frac{1}{4}} + w_2^2(k). \tag{6.22}$$

and the point spread function of the pixels assumed to be:

$$h(x, y, m, n, f(m,n)) = \left[\sin(x^2 + y^2)mnf^2(m,n)\right]^{\frac{1}{3}}. \tag{6.23}$$

For this model the mean of the initial state, x(0) was assumed to be 20 and the variance of it was chosen as 3.5. The process noise, w(k) and the observation noise v(k) were taken as zero mean Gaussian random variables with variances 1.25 and 2.5 respectively. All the simulations for this model were performed by using MATLAB version 7 running on a computer with an Intel Xeon – 3.4GHz processor and 4GB of RAM. For each simulation the estimation process for an image was performed for 250 times and the average of the results were calculated.

The original image and the blurred version of the original image according to the given models and statistics given above are presented in the figures 6-113 and 6-114 respectively.

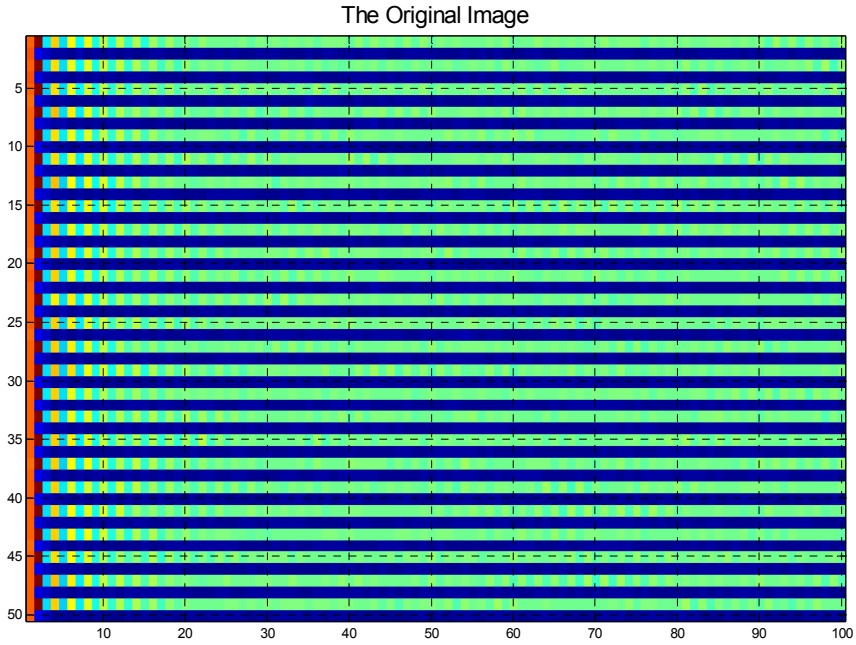

Figure 6-113 The Original Image according to the state transition equation

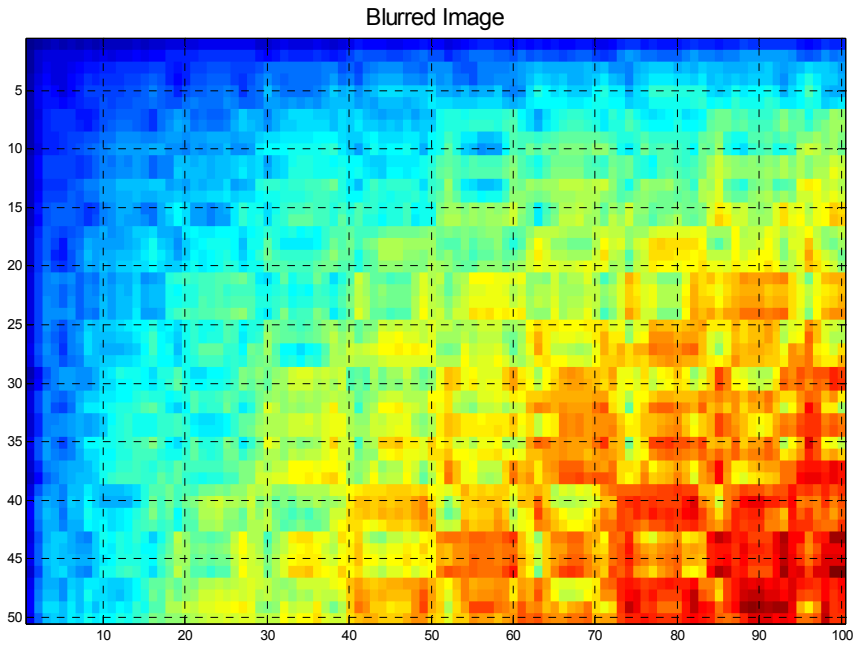

Figure 6-114 Result of the blurring function

6.5.1. Simulation 1

The results of the estimation algorithms with the parameters given below are as the following:

max_x_k_size	:9
gate_size	:0.1
no_of_sampling_points_x	:3
no_of_sampling_points_w	:3
Ns	:50

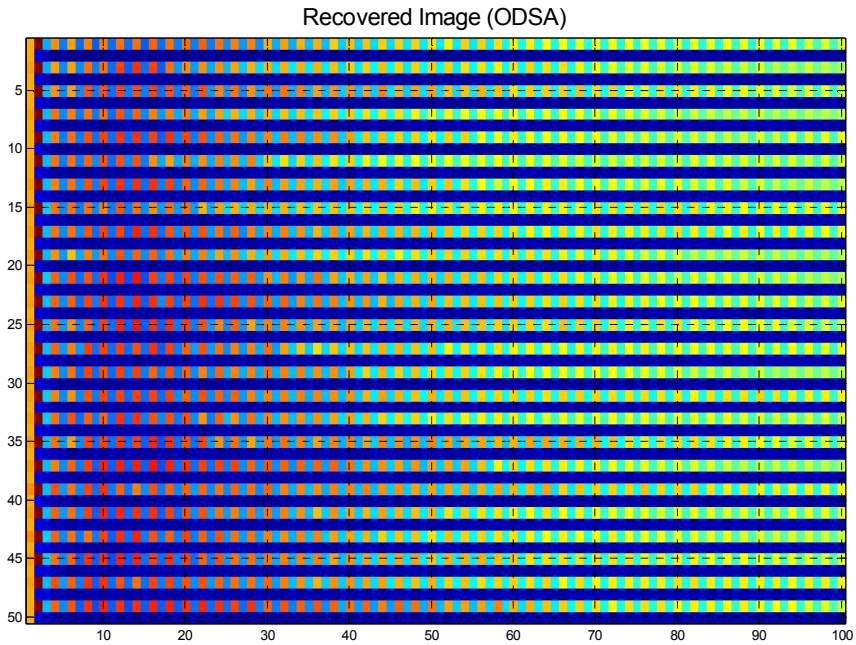

Figure 6-115 The Estimated Image by using the modified version of ODSA

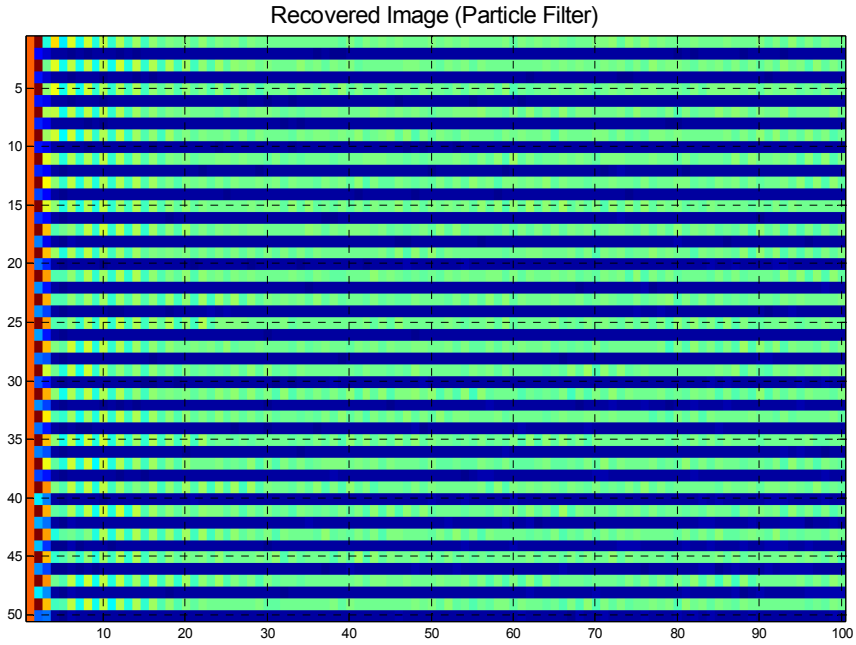

Figure 6-116 The Estimated Image by using the Bootstrap Filter Algorithm

The resulting mean_absolute_error for ODSA estimation is 3.5755 and the relative error is 39.19%. The time ODSA took for the estimation process is 7.6895 seconds per image.

When the Bootstrap Filter Algorithm was performed the mean_absolute_error became 1.9506 and the relative error was 21.38%. The processing time for the Bootstrap Filter was 3.2993 seconds per image.

6.5.2. Simulation 2

The second simulation for this model was performed with the following parameters:

max_x_k_size	:27
gate_size	:0.1
no_of_sampling_points_x	:3

no_of_sampling_points_w :3

Ns :100

The estimated images and the performance of the two algorithms are given below:

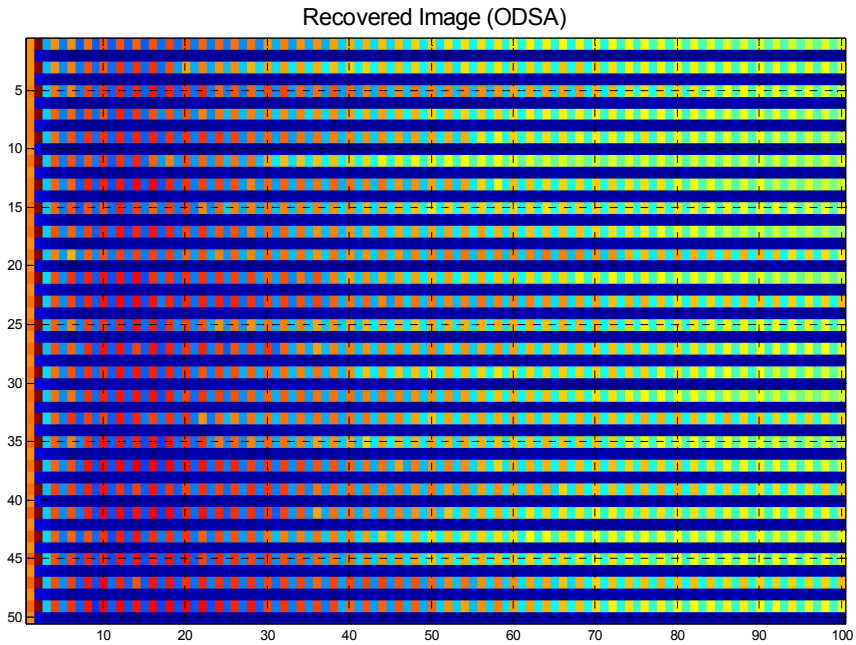

Figure 6-117 The Estimated Image by using the modified version of ODSA

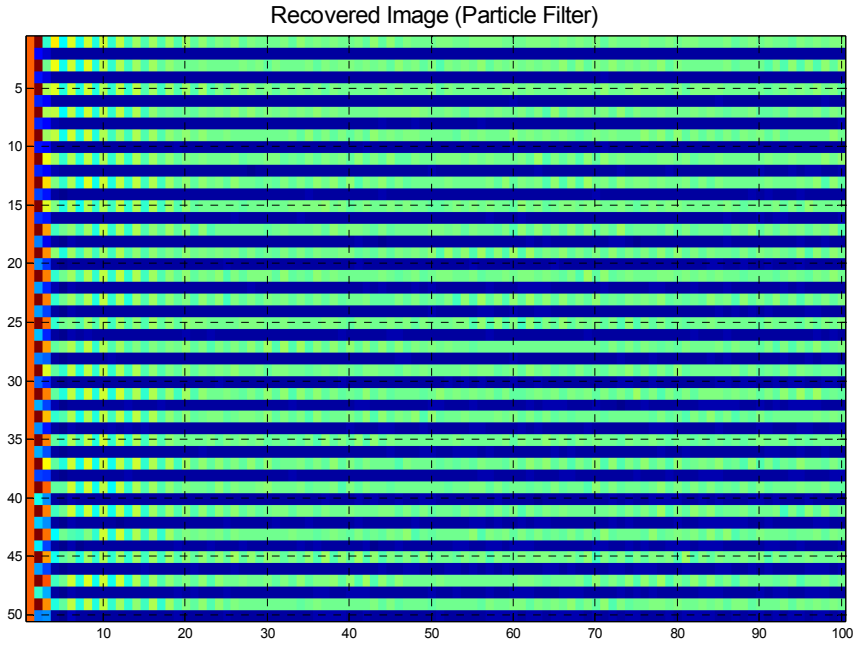

Figure 6-118 The Estimated Image by using the Bootstrap Filter Algorithm

The resulting mean_absolute_error for ODSA estimation is 3.7777 and the relative error is 41.41%. The time ODSA took for the estimation process is 27.8374 seconds per image.

When the Bootstrap Filter Algorithm was performed the mean_absolute_error became 1.8815 and the relative error was 20.62%. The processing time for the Bootstrap Filter was 7.8092 seconds per image.

6.5.3. Simulation 3

This time the parameters were set as the following:

max_x_k_size	:50
gate_size	:0.1
no_of_sampling_points_x	:3

no_of_sampling_points_w :3

Ns :150

With the parameters given above, the following results were obtained:

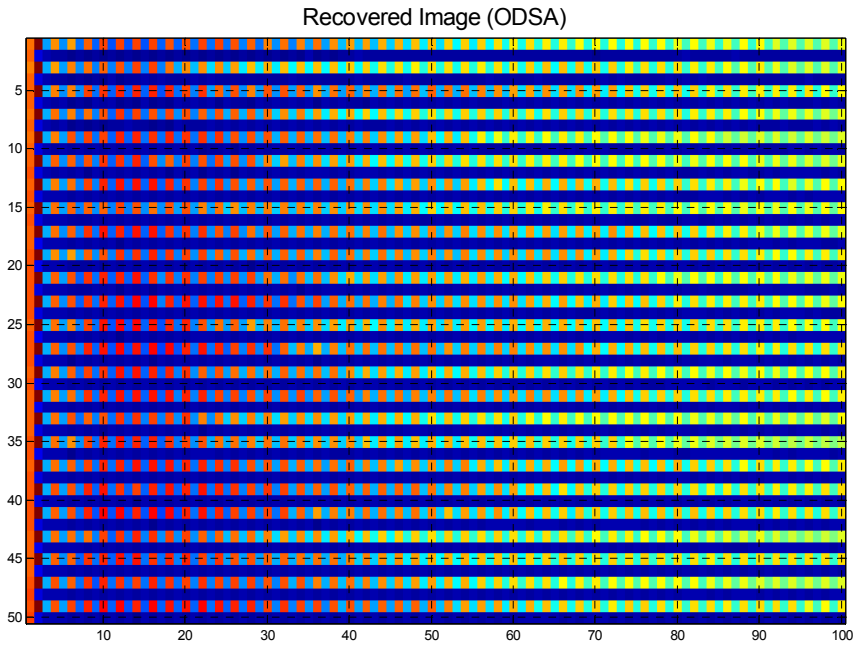

Figure 6-119 The Estimated Image by using the modified version of ODSA

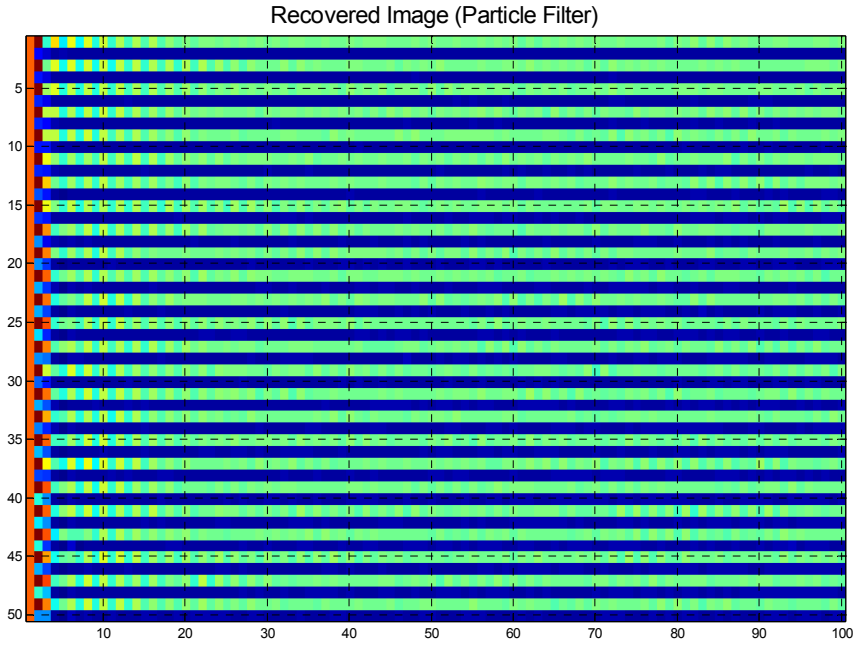

Figure 6-120 The Estimated Image by using the Bootstrap Filter Algorithm

The resulting mean_absolute_error for ODSA estimation is 3.8128 and the relative error is 41.81%. The time ODSA took for the estimation process is 68.6083 seconds per image.

With the Bootstrap Filter Algorithm the mean_absolute_error became 1.8612 and the relative error was 20.40%. The processing time for the Bootstrap Filter was 13.8453 seconds per image.

6.5.4. Simulation 4

For this simulation the parameters were changed to the following values:

max_x_k_size	:81
gate_size	:0.1
no_of_sampling_points_x	:3

no_of_sampling_points_w :3

Ns :250

The results are given below:

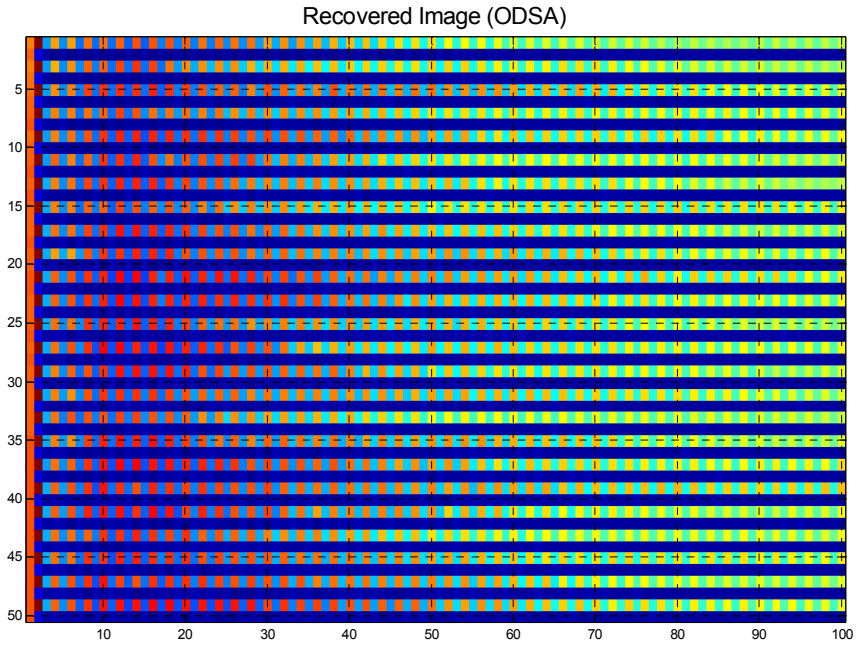

Figure 6-121 The Estimated Image by using the modified version of ODSA

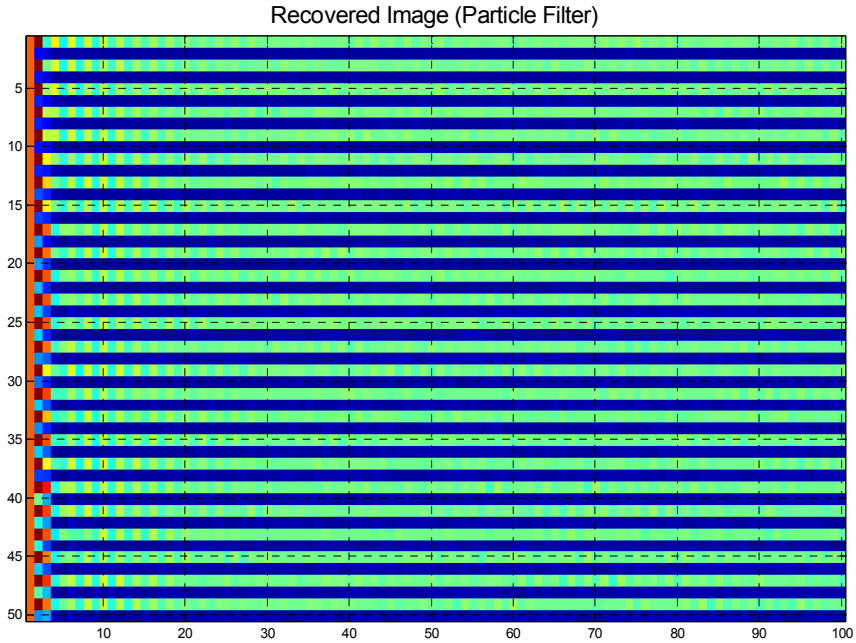

Figure 6-122 The Estimated Image by using the Bootstrap Filter Algorithm

For ODSA estimation, the resulting mean_absolute_error is 3.8399 and the relative error is 42.11%. The time ODSA took for the estimation process is 150.1296 seconds per image.

With the Bootstrap Filter Algorithm the mean_absolute_error became 1.8501 and the relative error was 20.29%. The processing time for the Bootstrap Filter was 31.2372 seconds per image.

6.5.5. Simulation 5

So far, the limit on the number of state nodes was increased to see its effect on the performance of the modified version of ODSA. This time the node limit was kept as the same but the gate size was varied. The simulation parameters for both of the algorithms are as the following:

max_x_k_size	:81
gate_size	:0.01
no_of_sampling_points_x	:3
no_of_sampling_points_w	:3
Ns	:500

The results with these parameters can be seen below:

Figure 6-123 The Estimated Image by using the modified version of ODSA

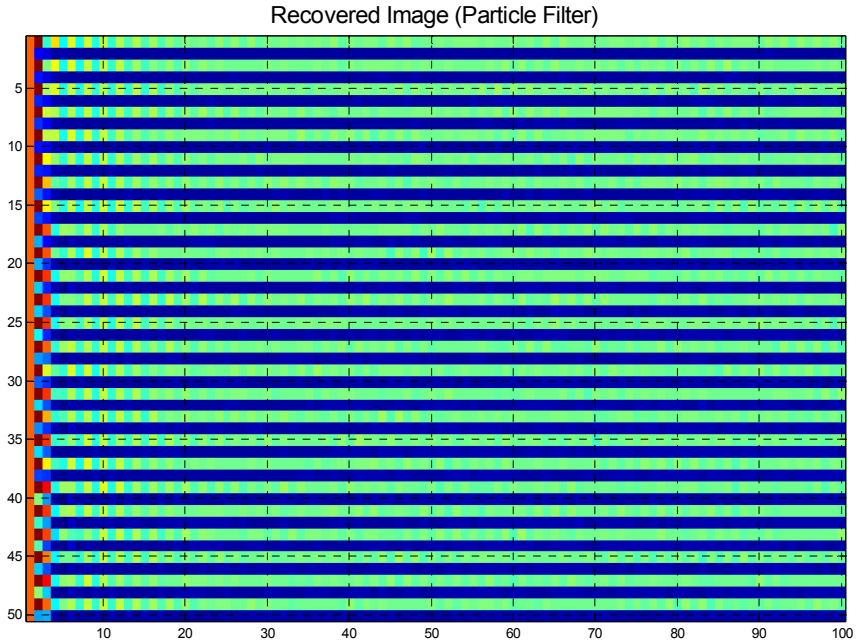

Figure 6-124 The Estimated Image by using the Bootstrap Filter Algorithm

The resulting mean_absolute_error for ODSA estimation is 3.7720 and the relative error is 41.35%. The time ODSA took for the estimation process is 130.6585 seconds per image.

When the Bootstrap Filter Algorithm was performed the mean_absolute_error became 1.8351 and the relative error was 20.12%. The processing time for the Bootstrap Filter was 106.7026 seconds per image.

6.5.6. Simulation 6

This time the following parameters were used for the image restoration process:

max_x_k_size	:81
gate_size	:0.25
no_of_sampling_points_x	:3

no_of_sampling_points_w :3

Ns :750

The results are given below:

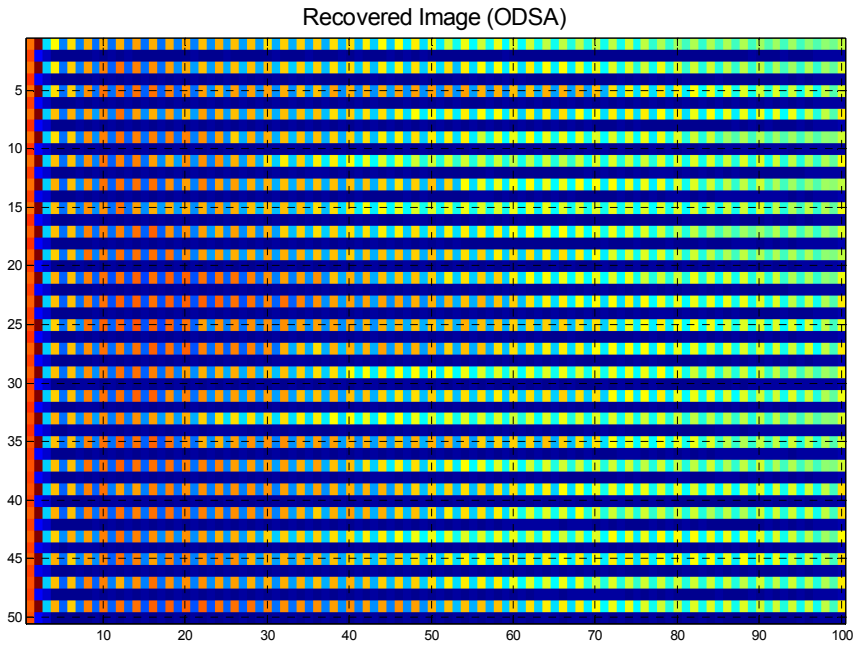

Figure 6-125 The Estimated Image by using the modified version of ODSA

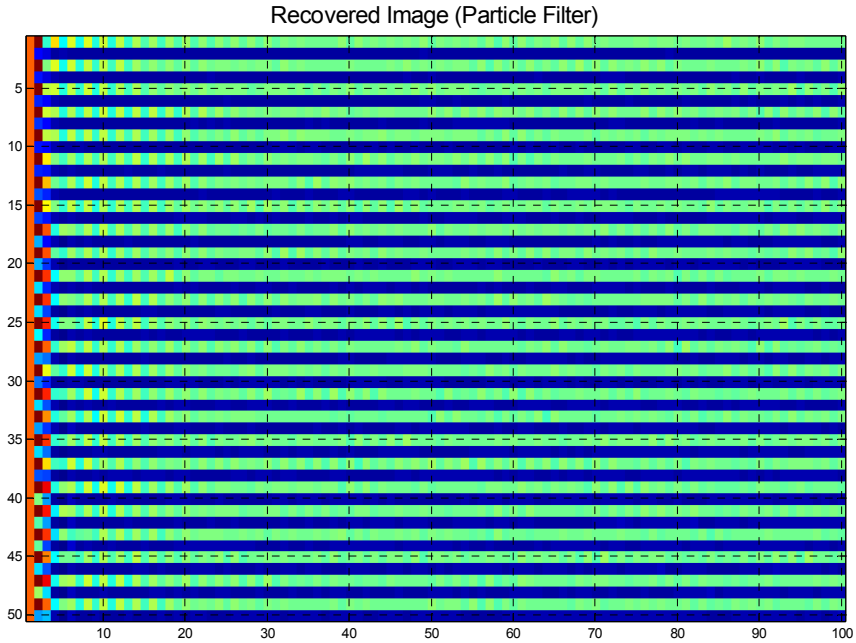

Figure 6-126 The Estimated Image by using the Bootstrap Filter Algorithm

For ODSA estimation, the resulting mean_absolute_error is 3.6577 and the relative error is 40.10%. The time ODSA took for the estimation process is 161.2155 seconds per image.

With the Bootstrap Filter Algorithm the mean_absolute_error became 1.8317 and the relative error was 20.09%. The processing time for the Bootstrap Filter was 227.8122 seconds per image.

6.5.7. Simulation 7

From this simulation on, the image restoration process was performed only by using the modified version of ODSA. The mentioned simulations were performed with different number of possible values of the initial state variable, x(0) and the disturbance noise variable, w(k). The parameters used are listed below:

max_x_k_size :50

gate_size :0.1

no_of_sampling_points_x :5

no_of_sampling_points_w :5

Figure 6-127 The Estimated Image by using the modified version of ODSA

The resulting mean_absolute_error is 3.3771 and the relative error is 37.01%. The time ODSA took for the estimation process is 375.0998 seconds per image in this case.

6.5.8. Simulation 8

The last simulation for this model was performed with the following parameters:

max_x_k_size :50

gate_size :0.1

no_of_sampling_points_x :7

no_of_sampling_points_w :7

Figure 6-128 The Estimated Image by using the modified version of ODSA

The resulting mean_absolute_error for this case is 3.3706 and the relative error is 36.93%. The time ODSA took for the estimation process is 1249.2000 seconds per image.

6.5.9. Summary of the Results

The graphics showing the error and process time performance of the estimation algorithms with respect to the change in the parameter values are given below:

Timing Performance Graphics:

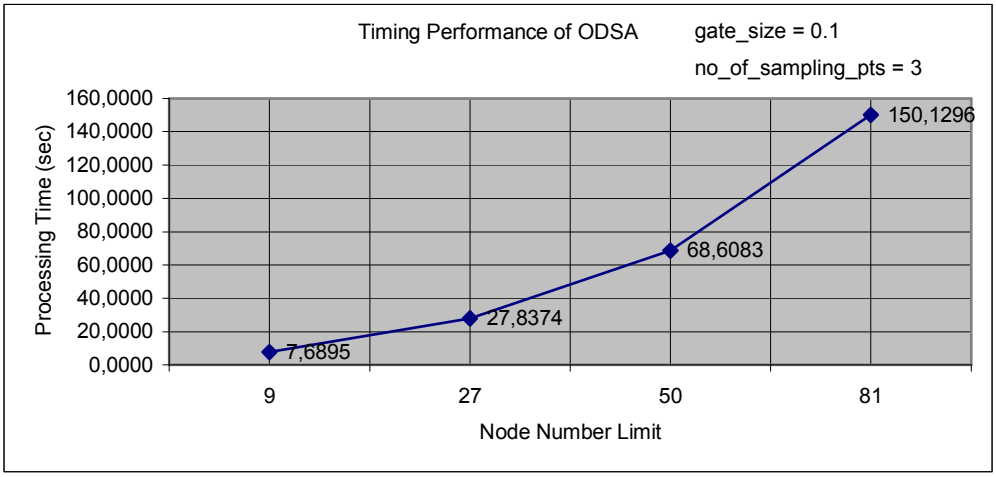

Figure 6-129 Timing Performance of ODSA with the parameters given on the graph

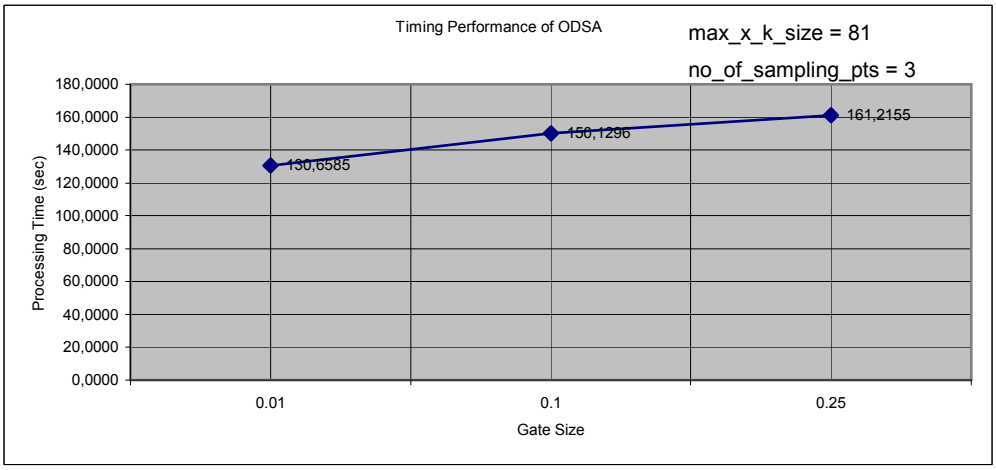

Figure 6-130 Timing Performance of ODSA with the parameters given on the graph

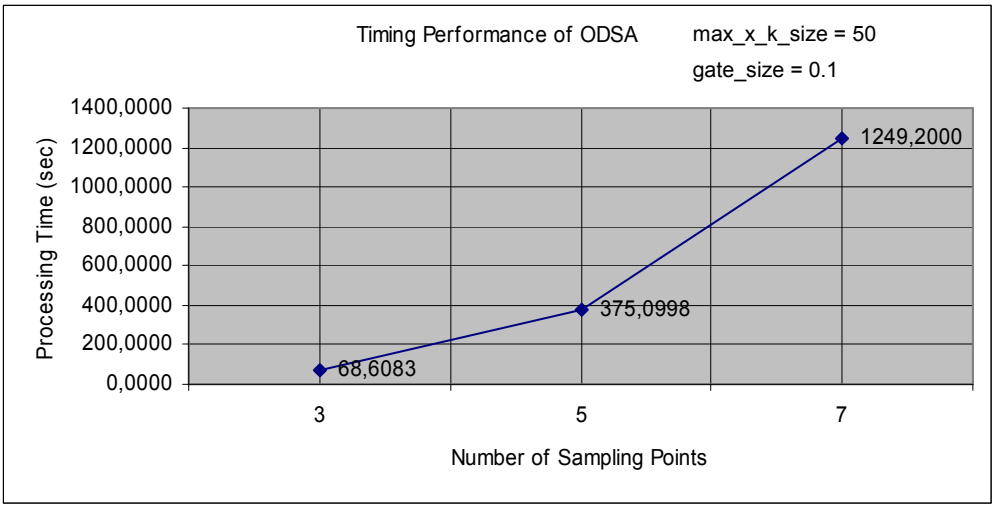

Figure 6-131 Timing Performance of ODSA with the parameters given on the graph

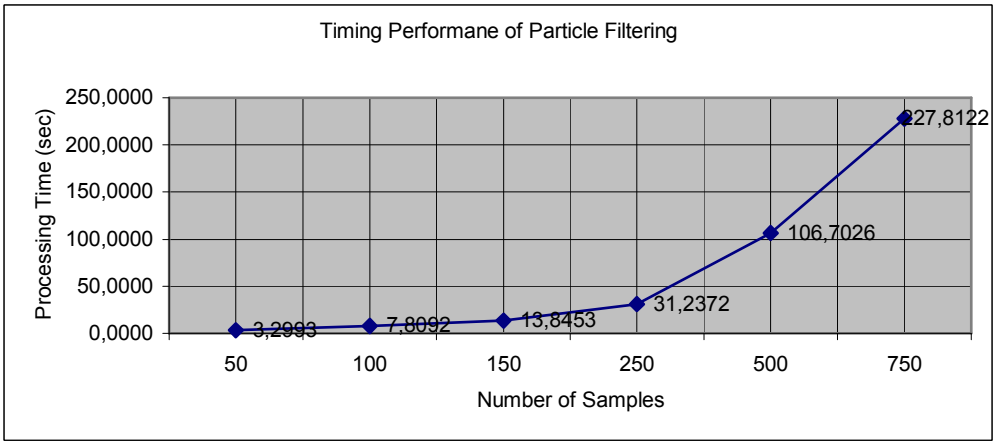

Figure 6-132 Timing Performance of the Bootstrap Filter Algorithm w.r.t. Ns

Error Performance Graphics:

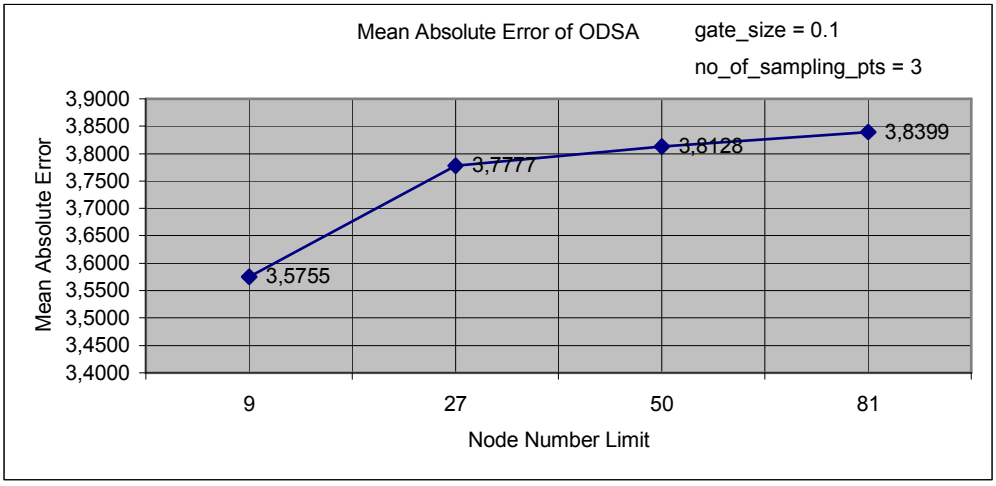

Figure 6-133 Mean absolute error per pixel for ODSA w.r.t. the parameters given on the graph

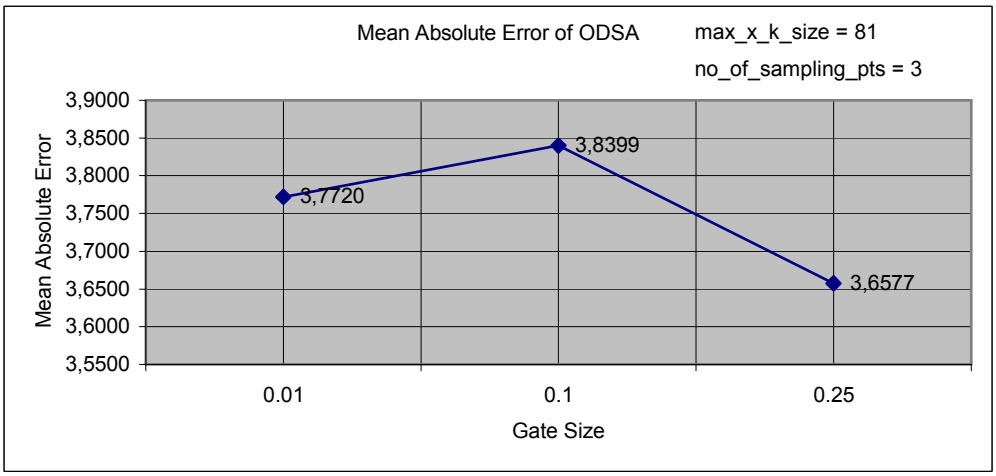

Figure 6-134 Mean absolute error per pixel for ODSA w.r.t. the parameters given on the graph

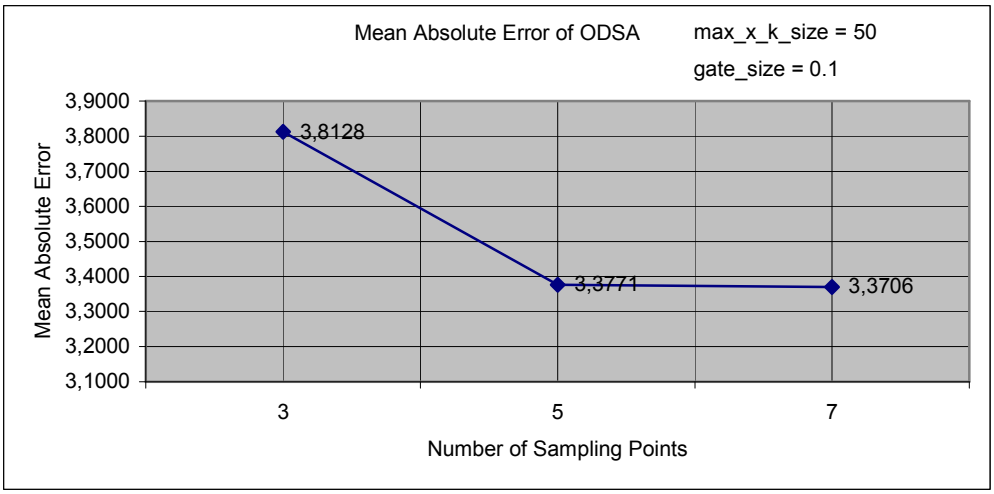

Figure 6-135 Mean absolute error per pixel for ODSA w.r.t. the parameters given on the graph

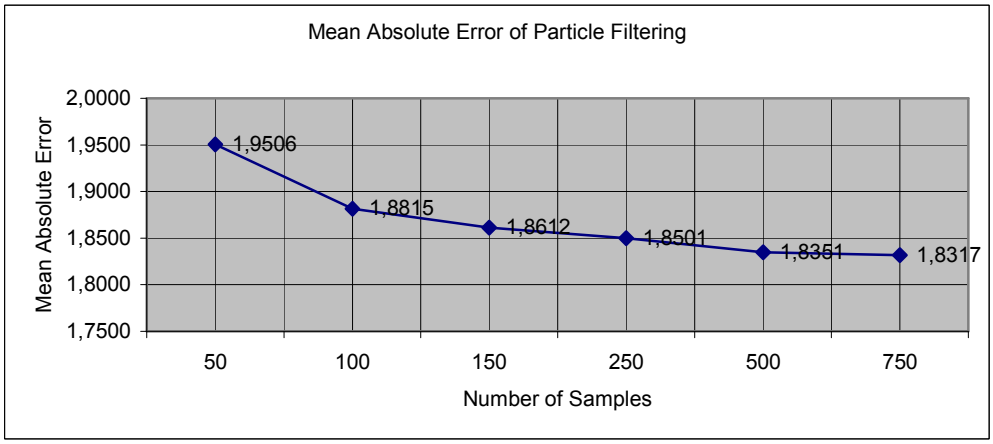

Figure 6-136 Mean absolute error per pixel for the Bootstrap Filter w.r.t Ns

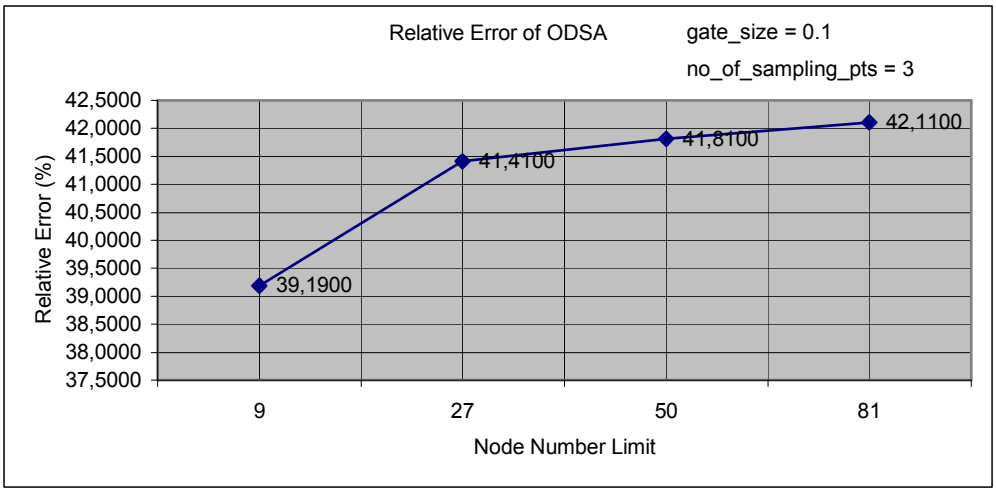

Figure 6-137 Relative error per pixel for ODSA w.r.t. the parameters given on the graph

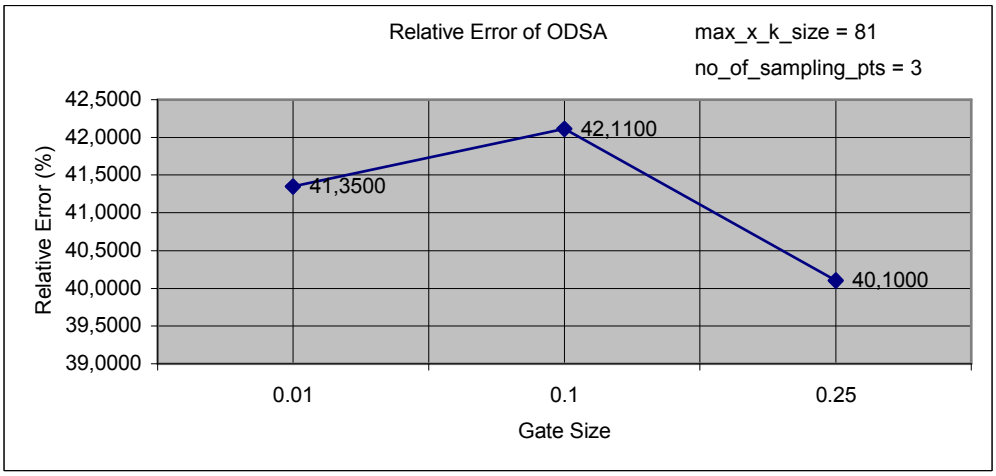

Figure 6-138 Relative error per pixel for ODSA w.r.t. the parameters given on the graph

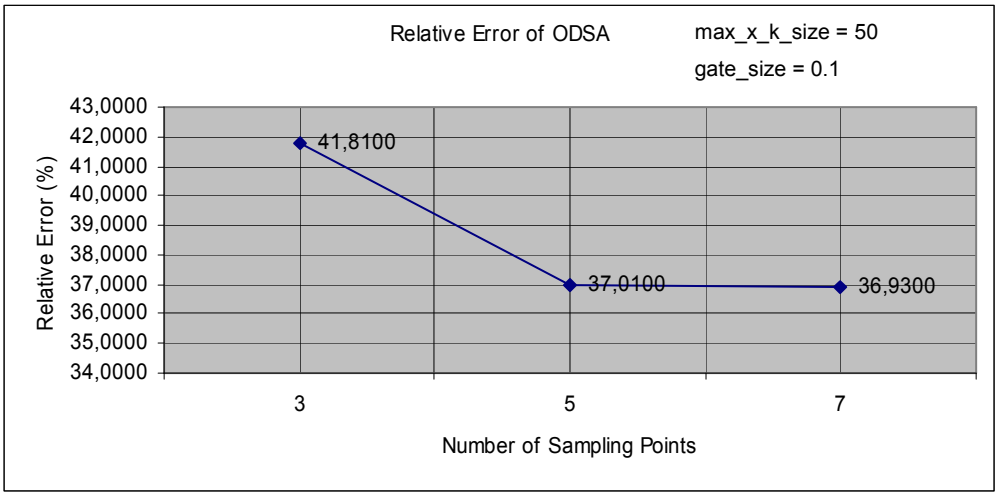

Figure 6-139 Relative error per pixel for ODSA w.r.t. the parameters given on the graph

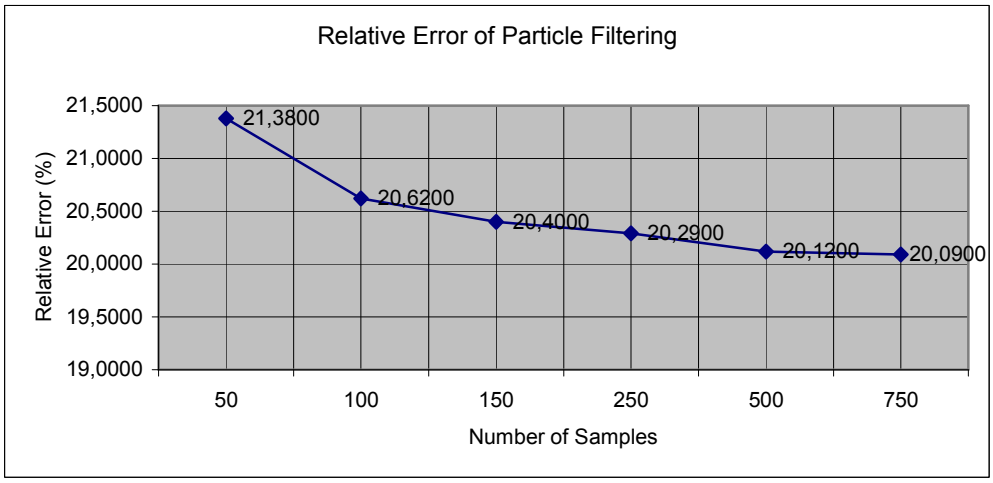

Figure 6-140 Mean absolute error per pixel for the Bootstrap Filter w.r.t Ns

The original image was generated randomly for each of the simulations. Therefore, relative_error value rather than mean_absolute_error should be used for the comparison of error performance of different simulations.

By using ODSA the best estimation for the original image has 39.19% relative error per pixel. This result was found in 7.6895 seconds for a single image on average. This simulation is also the fastest one. The best estimation obtained with the Bootstrap Filter has 20.09% relative error per pixel. This

result was obtained in 227.8122 seconds for a single image on average. The fastest estimation with the Bootstrap Filter Algorithm has been reached in 3.2993 seconds for a single image on average with 21.38% relative error per pixel.

CHAPTER 7

CONCLUSION

In this thesis, simulations for nonlinear image restoration by using two different methods; a modified version of the Optimum Decoding Based Smoothing Algorithm (ODSA) [1] and the Bootstrap Filter Algorithm [18], were performed.

The original ODSA[1] have unlimited nodes while drawing the trellis diagram. Therefore, the number of possible states at any time instant can diverge to infinity assuming the states do not intersect with each other periodically. The mentioned case needs a huge amount of memory and computational power; therefore the number of nodes of the trellis diagram at any time instant was limited to a predefined constant value. This modified version ODSA was again modified for two dimensional state estimation problems. The Bootstrap Filter Algorithm [18] was also adapted to the nonlinear image restoration problem.

For the restoration simulations, first an image, called original image or ideal image, wais created in strips with a width of two pixels by the state transition equations given in the simulations. Then the original images were blurred by the given observation equations given in the simulations. After the blurring operation, the modified version of the ODSA [1] and the Bootstrap Filter Algorithm were used to estimate the original images with varying algorithm parameters. The state transition and observation models could be linear or nonlinear, continuous or discontinuous; they could be any given functions.

The image restoration simulations were of Monte Carlo type. For five different models, image restoration simulations were performed for 250 times and average

results were calculated. According to the simulations, the error and timing performance of the algorithms are dependent not only on the algorithm parameters but also on the image and observation models. But it is clear that the overall performance of the Bootstrap Filter Algorithm [18] was better than the modified version of ODSA [1] for the models under consideration. Additionally, it can be seen clearly from the simulations that the error performance of the Bootstrap Filter can be improved by increasing the number of samples taken from the required PDF, i.e. Ns in the simulations. But this time, since the increase in the number of the samples creates an increase in the memory consumption and the number of computations needed, the processing time increases. Therefore, the optimum solution can be found with some trials by using the image and observation models of the application.

REFERENCES

[1] Kerim Demirbaş, "Information Theoretic Smoothing Algorithms for Dynamic Systems with or without Interference: Advances in Control and Dynamic Systems", Vol. XXI, Academic Press, pp. 175-295, 1984.

[2] Öner Balçıklı, "Nonlinear Image Restoration", A Master's Thesis, METU December 1999.

[3] H.C. Andrews, B. R. Hunt, "Digital Image Restoration", Prentice-Hall Inc, 1977.

[4] Pratt, William K., "Digital Image Processing", John Wiley & Sons Ltd., 1991.

[5] Aggelos K. Katsaggelos (Ed.), "Digital Image Restoration", Springer-Verlag, 1991.

[6] Arthur R. Weeks, Jr., "Fundamentals of Electronic Image Processing", IEEE Press,1996.

[7] Bindinganavle R. Suresh and B. A. Shenoi, "New Results in Two-Dimensional Kalman Filtering with Applications to Image Restoration", IEEE Transactions on Circuits and Systems, Vol. Cas-28, No. 4, pp. 307-319, April 1981.

[8] B. R. Hunt, "Digital Image Processing", Proceedings of the IEEE, Vol. 63, No. 4, pp. 693–708 April 1975.

[9] W. L. Rogers, K. S. Han, L. W. Jones, and W. H. Beiemaltes, “Application of a Fresnel zone plate to gamma-ray imaging,” J . Nucl. Med., Vol. 13, p. 612, 1972.

[10] H. J. Trussel, “A Priori Knowledge in Algebraic Reconstruction Methods”, Advances in Computer Vision and Image Processing, vol.1, T. S. Huang, Ed. JAI 1984.

[11] M. E. Zervakis, A. N. Venetsanopoulos, “Iterative Least Squares Estimators in Nonlinear Image Restoration”, IEEE Transactions on Signal Processing, Vol. 40, No. 4, April 1992.

[12] B. R. Hunt, “Bayesian Methods in Nonlinear Digital Image Restoration”, IEEE Trans. Comput., Vol.C-26, No. 3, March 1977.

[13] S. Geman, D. Geman, “Stochastic Relaxation, Gibbs Distributions, and the Bayesian Restoration of Images”, IEEE Trans. Patt. Anal. Machine Intell., Vol. PAMI-6, No.6, November 1984.

[14] A. V. Oppenheim, R. W. Schafer and T.G. Stockham, “Nonlinear Filtering of Multiplied and Convolved Signals”, Proceedings of the IEEE, Vol. 56, pp.1264-1291, August 1968.

[15] R. W. Fries, J. W. Modestino, “Image Enhancement by Stochastic Homomorphic Filtering”, IEEE Trans. Acoust., Speech, Signal Processing, Vol. ASSP-27, No.6, December 1979.

[16] V. S. Frost, J. A. Stiles, K. S. Shanmugam, J. C. Holtzman, S. A. Smith, “An Adaptive Filter for Smoothing Noisy Radar Images”, Proceedings of the IEEE, Vol. 69, No.1, January 1981.

[17] A. M. Apostol, “Mathematical Analysis”, Addison-Wesley, Reading, Massachusetts, 1958

[18] Arnaud Doucet, Nando de Freitas, Neil Gordon, “An Introduction to Sequential Monte Carlo Methods” in A. Doucet, N. de Freitas, N. Gordon (Eds.), “Sequential Monte Carlo Methods in Practice”, Springer-Verlag New York Inc., 2001

[19] N. J. Gordon, D. J. Salmond, A. F. M. Smith, “Novel Approach to Nonlinear / Non-Gaussian Bayesian State Estimation”, IEE Proceedings-F, Vol. 140, No.2, April 1993

[20] B. P. Carlin, N. G. Polson, D.S. Stoffer, “A Monte Carlo Approach to Nonnormal and Nonlinear State Space Modelling”, J. Amer. Statistical Assoc.,87, pp. 493-500, 1992

[21] M. West, “Modelling with Mixtures (with discussion)”, in J. M. Bernardo, J. O. Berger, A. P. Dawid, A. F. M. Smith (Eds) “Bayesian Statistics 4”, Oxford University Press, pp. 503-524, 1992

[22] P. Muller, “Monte Carlo Integration in General Dynamic Models”, Contemp. Math., 115, pp. 145-163, 1991

APPENDIX

APPROXIMATION OF AN ABSOLUTELY CONTINUOUS RANDOM VECTOR BY A DISCRETE RANDOM VECTOR [1]

Let n be a given positive integer and let D^m be the set of distribution functions of all m x 1 discrete random vectors with n possible values, where superscript m stands for the dimensionality of random vectors. Then the problem of approximating an absolutely continuous m x 1 random vector X^m with distribution function $F_{X^m}(.)$ by an m x 1 discrete random vector with n possible values is to find a distribution function $F_{Y_0^m}(.) \in D^m$, which minimizes the objective function $J(.)$ over the set D^m:

$$J\left(F_{Y_0^m}(.)\right) = \min_{F_{Y^m}(.) \in D^m} J\left(F_{Y^m}(.)\right). \tag{A.1}$$

where

$$J\left(F_{Y^m}(.)\right) \triangleq \int_{R^m} \left[F_{X^m}(a) - F_{Y^m}(a)\right]^2 da, \qquad F_{Y^m}(.) \in D^m. \tag{A.2}$$

Note that the integration is performed over the m dimensional Euclidean space R^m. The discrete random vector defined by $F_{Y_0^m}(.)$ is referred to as the optimum discrete random vector approximating the random vector X^m. Here, the approximation of an absolutely continuous random variable X with distribution $F_X(.)$ by a discrete random variable with n possible values is considered. The necessary conditions that the optimum discrete random variable approximating X must satisfy are obtained. Finally, discrete random variables approximating normal random variables are obtained.

Let us now state two theorems and define some symbols which are used. The proofs of the theorems are given in Ref. [17].

Theorem A.1 [17]

Let $f(y) \triangleq f(y_1, y_2, ..., y_l)$ be a real valued function on an open set Γ of R^l, and let $f(y)$ have finite partial derivatives $\frac{\partial f(y)}{\partial y_k}$, k=1, 2, …, l at each point of Γ. If $f(y)$ has a local minimum at the point $y_0 \triangleq (y_{1,0}, y_{2,0}, ..., y_{l,0})$ in Γ, then $\left.\frac{\partial f(y)}{\partial y_k}\right|_{y=y_0} = 0$ for each k=1, 2, …, l.

Theorem A.2 [17]

Let $f(y) \triangleq f(y_1, y_2, ..., y_l)$ be a real valued function on an open set Γ of R^l, and let $f(y)$ have continuous second-order partial derivatives on Γ. Let $y_0 \triangleq (y_{1,0}, y_{2,0}, ..., y_{l,0})$ be a point of Γ for which $\left.\frac{\partial f(y)}{\partial y_k}\right|_{y=y_0} = 0$ for each k=1, 2, …, l. Assume that the determinant $G \triangleq \det\left\{\left[\nabla^2 f(y)\right]_{y=y_0}\right\} \neq 0$, where

$$\left[\nabla^2 f(y)\right]_{ij} \triangleq \frac{\partial^2}{\partial y_i \partial y_j} f(y).$$

Let G_{l-k} be the determinant obtained from G by deleting the last k rows and columns. If the l numbers G_1, G_2, …, G_l are all positive, then $f(y)$ has a local minimum at y_0.

We now define D as the set of distribution functions of all discrete random variables with n possible values, where n is a given positive integer. We next define S as the set of all stepfunctions with n steps. A stepfunction g(.) with n steps is, by definition, a function with n + 1 possible values in the real line R such that g(.) is zero at -∞ and one at +∞; the set of numbers which are mapped by g(.) to a chosen possible value is an interval in R, the intervals corresponding to the n + 1possible values are nonoverlapping, and the union of these intervals is the real line, that is,

$$S \triangleq \{g(x): g(x) = 0 \text{ for } x < y_1;$$
$$g(x) = P_i \text{ for } y_i \le x < y_{i+1}, P_i \in (-\infty, \infty);$$
$$g(x) = 1 \text{ for } x \ge y_n; y_{i+1} > y_i, y_i \in (-\infty, \infty);$$
$$i = 1,2,...,n-1\}.$$

In order to find the optimum discrete random variable with n possible values that approximates an absolutely continuous random variable X with distribution function $F_X(.)$, we must find a distribution function $F_{Y_0}(.)$ which minimizes the objective function $J(.)$ over the set D:

$$J(F_{Y_0}(.)) = \min_{F_Y(.) \in D} J(F_Y(.)), \tag{A.3}$$

$$= \min_{g(.) \in S} J(g(.)). \tag{A.4}$$

where

$$J(F_Y(.)) \triangleq \int_{-\infty}^{\infty} [F_X(a) - F_Y(a)]^2 \, da, \tag{A.5}$$

The equality in Eq. (A.4) follows from the following arguments. Let a stepfunction $g_0(.) \in S$ minimize $J(.)$ over the set S; since the distribution function $F_X(.)$ is nondecreasing, $g_0(.)$ must be nondecreasing; hence it is a nondecreasing stepfunction whose range changes from zero to one; therefore $g_0(.) \in D$. Thus the aim is to find a aim is to find a stepfunction $g_0(.) \in S$ which minimizes the objective function $J(.)$ over S. That is we would like to minimize the following function over $y_i \in (-\infty, \infty)$ and $P_j \in (-\infty, \infty)$ (where i=1, 2, …, n; j=1, 2, …, n-1):

$$J(g(.)) = \int_{-\infty}^{y_1} F_x^2(a) da + \int_{y_1}^{y_2} [F_x(a) - P_1]^2 da + \int_{y_2}^{y_3} [F_x(a) - P_2]^2 da + ...$$
$$+ \int_{y_{n-1}}^{y_n} [F_x(a) - P_{n-1}]^2 da + \int_{y_n}^{\infty} [F_x(a) - 1]^2 da, \tag{A.6}$$

It follows from Theorem A.1 that if $g_0(x)$, which is defined by

$$g_0(x) = \begin{cases} 0, & x < y_{1,0}, \\ P_{i,0}, \text{ if} & y_{i,0} \le x \le y_{i+1,0}, \\ 1, & x \ge y_{n,0} \end{cases} \qquad \text{i=1, 2, …, n-1} \qquad \text{(A.7)}$$

is a stepfunction which minimizes equation (A.6), this must satisfy the following set of equations:

$$\begin{aligned} &P_{1,0} = 2F_x(y_{1,0}); \\ &P_{i,0} + P_{i+1,0} = 2F_x(y_{i+1,0}), \qquad \text{i=1, 2, 3, …, n-2;} \\ &1 + P_{n,0} = 2F_x(y_n); \end{aligned} \qquad \text{(A.8)}$$

$$P_{i,0}(y_{i+1,0} - y_{i,0}) = \int_{y_{i,0}}^{y_{i+1,0}} F_x(a)da, \qquad \text{i=1, 2, …, n-1}$$

Using equation (A.8) and Theorem A.2, the discrete random variables (with n possible values where n=1, 2, …, 8) which approximate the normal random variable with zero mean and unit variance have been numerically obtained and are tabulated in Table A.1.

Table A-1. Discrete Random Variables Approximating the Gaussian Random Variable with Zero Mean and Unit Variance

number of possible values of y_0 *	n possible values and corresponding probabilities of y_0								
	i **	1	2	3	4	5	6	7	8
n=1	$y_{i,0}$	0.000							
	$P_{i,0}$	1.000							
n=2	$y_{i,0}$	-0.675	0.675						
	$P_{i,0}$	0.500	0.500						
n=3	$y_{i,0}$	-1.005	0.000	1.005					
	$P_{i,0}$	0.315	0.370	0.315					
n=4	$y_{i,0}$	-1.219	-0.355	0.355	1.219				
	$P_{i,0}$	0.223	0.277	0.277	0.223				
n=5	$y_{i,0}$	-1.376	-0.592	0.000	0.592	1.376			
	$P_{i,0}$	0.169	0.216	0.230	0.216	0.169			
n=6	$y_{i,0}$	-1.499	-0.767	-0.242	0.242	0.767	1.499		
	$P_{i,0}$	0.134	0.175	0.191	0.191	0.175	0.134		
n=7	$y_{i,0}$	-1.599	-0.905	-0.423	0.000	0.423	0.905	1.599	
	$P_{i,0}$	0.110	0.145	0.162	0.166	0.162	0.145	0.110	
n=8	$y_{i,0}$	-1.683	-1.018	-0.567	-0.183	0.183	0.567	1.018	1.683
	$P_{i,0}$	0.093	0.123	0.139	0.145	0.139	0.139	0.123	0.093

*: y_0, the discrete random variable with n possible values $y_{1,0}$, $y_{2,0}$, …, $y_{n,0}$, which approximates the normal random variable with zero mean and unit variance.

**: $y_{i,0}$, the ith possible value of y_0 : $P_{i,0} \triangleq \Pr ob\{y_0 = y_{i,0}\}$.

Let y_0 be the optimum discrete random variable with n possible values $y_{1,0}$, $y_{2,0}$, …, $y_{n,0}$ which approximate the normal random variable with zero mean and unit variance, and let $P_{i,0}$ be defined by $\Pr ob\{y_0 = y_{i,0}\}$. Let z_0 be the optimum discrete random variable with n possible values $z_{1,0}$, $z_{2,0}$, …, $z_{n,0}$ which approximates the

normal random variable with mean μ and variance σ^2; and let $p'_{i,0}$ be defined by $\Pr ob\{z_0 = z_{i,0}\}$. By equation A.8, it can easily be verified that

$$z_{i,0} = \sigma y_{i,0} + \mu, \; p'_{i,0} = p_{i,0}, \quad \text{i=1, 2, …, n.} \tag{A.9}$$