A BAYESIAN APPROACH TO TEXTILE DEFECT DETECTION PROBLEM AND A COMPARATIVE ANALYSIS

by

Abdullah Yakın

B.S., Electrical and Electronics Engineering, Boğaziçi University, 2007

Submitted to the Institute for Graduate Studies in
Science and Engineering in partial fulfillment of
the requirements for the degree of
Master of Science

Graduate Program in Systems and Control Engineering

Boğaziçi University

2010

ACKNOWLEDGEMENTS

I would like to express my sincere gratitude to my supervisor, Prof. Ayşın Baytan Ertüzün, for her invaluable guidance and help during the preparation of this dissertation. I would like to mention her patience, giving me inspiration and hope when I was stuck at dead-ends.

I am grateful to Prof. Aytül Erçil for her friendship, guidance and support during my graduate education. Without her directions, it would be much more difficult to accomplish this study.

Also I should thank my friends Mustafa Emre Kırcalar, Mehmet Maden and Caner Korkmaz for their assist during the stressful period of work.

Last but not least, I would like to express my deepest gratitude to my dear family, who has supported me with their endless love and understanding.

ABSTRACT

A BAYESIAN APPROACH TO TEXTILE DEFECT DETECTION PROBLEM AND A COMPARATIVE ANALYSIS

The purpose of this thesis was to detect the defects on textile fabric images using the particle filters. We approached the problem from a Bayesian perspective and represented the model in a state space formulation. To describe the state space formulation; the texture models such as linear, 2-D linear and Markov Random Field models and the noise types like Gaussian, mixture of Gaussian and alpha-stable noise are investigated to find the best representation that is appropriate for our textile images. The implementation results are compared with Kalman, Extended Kalman and Unscented Kalman filters. Finally time and performance analysis of the filters is given.

ÖZET

TEKSTİL HATALARININ TESPİTİ PROBLEMİNE BAYESÇİ BİR YAKLAŞIM VE KARŞILAŞTIRMALI BİR ANALİZ

Bu tezde, tekstil doku imgelerindeki hataların, parçacık filtresi ile tespit edilebilmesi amaçlanmaktadır. Bunun için probleme Bayesçi bir perspektiften yaklaştık ve modelimizi durum uzayı formülasyonu ile ifade ettik. Durum uzayı formülasyonunu tanımlayabilmek için de; doğrusal, iki boyutlu doğrusal ve Markov rassal alan gibi örüntü modellerini; Gauss, Gauss karışımı ve alfa-durağan gibi de gürültü çeşitlerini doku imgelerini en iyi şekilde temsil edebilmek için inceledik. Elde ettiğimiz sonuçları; diğer Kalman, genişletilmiş Kalman ve kokusuz Kalman filtelerinden elde edilen sonuçlarla karşılaştırdık. Son olarak da, filtrelerin zaman ve performans analizini verdik.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF SYMBOLS / ABBREVIATIONS

a	Markov Random Field clustering parameter
a_n	One dimensional autoregressive coefficient
$a_{m,n}$	Two dimensional autoregressive coefficient
$\hat{\mathbf{a}}$	Vector of AR coefficients
A^T	Transpose of matrix A
b_j	Markov Random Field clustering parameter
c_i	Mixture proportion in Mixture of Gaussian noise
d_i	Distance between the true and the test state vector
D	Distance vector consisting of $d_i's$
D_m	Median of the distance vector
D_u	Median of the distances bigger than D_m
D_l	Median of the distances smaller than D_m
f	State model function
F	Linear state model matrix
$\hat{\mathbf{F}}$	First order Taylor series expansion of function **f**
h	Observation model function
H	Linear observation model matrix
$\hat{\mathbf{H}}$	First order Taylor series expansion of function **h**
k	Time index k
K	Kalman gain matrix
l	Likelihood in Markov Random Field parameter estimation
$l(i)$	Likelihood of the *i*'th coding in Markov Random Field parameter estimation
n	Observation model noise
n_c	Number of clusters in Mixture of Gaussian noise
N	Number of particles

$\mathcal{N}(x;\mu,\sigma^2)$	Normal distribution of argument x with mean μ and variance σ^2
$p(.)$	Probability density function
$p(.\|.)$	Conditional probability density function
$\mathbf{P}_x$	Covariance of sigma points in unscented transformation
$\mathbf{P}_y$	Covariance of new transformed points in unscented transformation
$q(.)$	Proposal density
$\mathbf{Q}$	State model covariance matrix
$\mathbf{R}$	Observation model covariance matrix
T	Markov Random Field parameter
Th	Threshold value
$\mathbf{v}$	State model noise
w	Particle weights
W	Weight of sigma points in unscented transformation
$\mathbf{x}_k$	State vector at time k
$\mathbf{x}_k^i$	Sample value at time k
$\mathbf{x}_{true}$	Estimated true state vector
$\mathbf{x}_{test}$	Estimated test state vector
$\bar{\mathbf{x}}$	Mean vector of $\mathbf{x}$
$\hat{x}$	Predicted value of x
$\mathbf{X}$	Calculated sigma points in unscented transformation
$\mathbf{y}_k$	Observation vector at time k
$\mathbf{Y}$	Transformed sigma points in unscented transformation
α	Characteristic exponent of an alpha stable distribution
β	Symmetry parameter of an alpha stable distribution
γ	Scale parameter of an alpha stable distribution
δ	Location parameter of an alpha stable distribution
η	Ridge parameter
θ	Robust scale parameter

κ	Scaling parameter in unscented transformation
λ	Scaling parameter for threshold specification
$\mathbf{\Lambda}$	Diagonal matrix in iteratively reweighted least squares algorithm
μ	Mean of the Gaussian distribution
$\rho(x)$	Objective function in iteratively reweighted least squares algorithm
σ^2	Variance of the Gaussian distribution
$\sum$	Covariance matrix
τ	Auxiliary particle filter point estimates
$\varphi(t)$	Characteristic function of an alpha stable distribution

1-D	One dimensional
2-D	Two dimensional
αS	Alpha Stable
AR	Autoregressive
ASIR	Auxiliary Sampling Importance Sampling
cdf	Cumulative distribution function
EKF	Extended Kalman Filter
i.i.d	independently and identically distributed
IRLS	Iteratively Reweighted Least Squares
LS	Least Squares
MLE	Maximum likelihood estimation
MoG	Mixture of Gaussian
MRF	Markov Random Field
pdf	Probability density function
$S\alpha S$	Symmetric Alpha Stable
SIR	Sampling Importance Resampling
SIS	Sequential Importance Sampling
UKF	Unscented Kalman Filter

1. INTRODUCTION

Automated visual control of products is one of the most crucial subjects in almost all industry branches in our modern world. One of those critical branches that can affect all human beings' daily life is textile industry. The production of first quality textile fabric in textile industry makes the post processing meaningful since it represents a surface that is totally free of major defects. A major defect that is missed in this step will increase the cost and decrease the efficiency in the next stages following this process.

As the production speeds increase, the importance of automated visual inspection also increases because manufacturers must be able to identify the defects, locate their source, and make the necessary corrections in less time [1]. This leads to more awareness of research and development engineering since an efficient performance rate in production lines should be shown in a limited time period. So, many production managers give support to related engineers to focus on texture analysis.

Texture analysis plays an important role in the automated visual inspection of surfaces. There have been number of works for texture defect detection in the literature. Chen and Jain [2] used a defect detection approach in which they first threshold the textured image using histogram analysis and then defining several statistics, the defects are identified. Aras *et al.* [3] defined higher order statistical feature sets and using a Mahalanobis distance classifier, defects on textural images of textile fabrics are detected. Amet *et al.* [4] combined concepts from wavelet theory and co-occurrence matrices for defection of defects. Detection of the defects is performed by decomposing the gray level image into sub-bands and classifying the partitioned sub-windows according to their extracted co-occurrence features. Similarly, Yang *et al.* [5] used wavelet transform for defect detection but they developed adaptive wavelets adapting to the detection of the fabric defects instead of predetermining a wavelet. Kumar and Pang [6] proposed to use FIR linear filters with optimized energy separation to find defects on fabric images.

In this work, we approach the texture defect detection problem from a Bayesian perspective using a state space formulation. State space formulation gives us a relation

between the states changing over time which are unobservable and a relation between the state and the observation that is made on the system. We define these relations in the models as state model and observation model and they are very useful to describe the system dynamics.

Our motivation is to form the state space model accurately by finding the characteristics of the textile images as well as possible. Constructing the state space model is a vital step to approach the problem from a Bayesian perspective.

Texture defect detection using a Bayesian approach was first proposed by Basibuyuk *et al.* [7] and it was assumed that the state model and the observation model were both linear and Gaussian. They applied particle filter in this environment and obtained results comparable to Kalman filtering as expected, the results were promising for further investigation in nonlinear non-Gaussian environments.

The main contribution of this dissertation is to expand this approach by applying the particle filter to state space models that include nonlinearity, non-Gaussianity or both. While analyzing the nonlinearity of the system, we investigate several texture description models and check them if they are appropriate for filtering purposes. Similarly various noise descriptions are revealed to find the best noise term that fits well to the textile fabric images.

Another contribution of this work is to make it possible to evaluate the differences of Kalman filter and particle filter for textile defect detection problem. For this purpose, Kalman filter and its extensions are explained to make the subject more clear and understandable. Kalman filter is the optimum filter for linear and Gaussian environments so we compared its results with the ones obtained from the particle filter to understand that if the environment is really linear and Gaussian. Extended Kalman and unscented Kalman filter are also discussed to show how the performance of the Kalman filter can be increased in nonlinear environments.

Also it is aimed to make the implementation as fast as possible to run it on a real time machine vision system. Time and performance analyses are done and the optimum filter is proposed for textile fabric defect detection problem.

After the implementations, it is concluded that the dynamics of the textile fabric images are suitable to represent them in a linear and Gaussian state space model. For this system, Kalman filter gives the optimum results both in achieved performance and operation time.

The organization of the dissertation is as follows: In section 2, we will give a theoretical background for state space formulation and Bayesian approach; explain most known filter types, texture models and noise types. In section 3, we will demonstrate our experimental results and share the approaches we have used. In section 4, we will discuss the results obtained from the implementations and try to give possible directions for future work.

2. THEORETICAL BACKGROUND

In this chapter, we introduce the theoretical basis of our work. We try to model our system in a state space formulation and then give a probabilistic approach to this formulation. In this context, Kalman filter and its extensions, particle filters, various noise descriptions and texture models are explained.

2.1. State Space Formulation

One of the most basic definitions in control theory is the state space formulation. It gives us an ability to find the states using a sequence of noisy measurements made on the system. Many fields of science require this closed form formulation to estimate the states in a dynamic environment.

If we can find the state vector of the system, it gives us relevant information describing the system under investigation. State space formulation gives us two models to find the state vector. First model, the state model, gives us a relation between the states changing over time which are unobservable and the second model, the observation model, gives us the relationship between the state and the observation made on the system.

We describe the state model as

$$\mathbf{x}_k = \mathbf{f}_k(\mathbf{x}_{k-1}, \mathbf{v}_{k-1}) \tag{2.1}$$

where $\mathbf{f}_k$ is the function of the evolution of the states and $\mathbf{v}_{k-1}$ is the independently and identically distributed (i.i.d) state noise of the system. Similarly the observation model can be defined as

$$\mathbf{y}_k = \mathbf{h}_k(\mathbf{x}_k, \mathbf{n}_k) \tag{2.2}$$

where $\mathbf{h}_k$ is the function of the relationship between the state and the observation and $\mathbf{n}_k$ is the i.i.d observation noise of the system.

In this formulation, each of the functions $\mathbf{f}_k$ and $\mathbf{h}_k$ can be linear or nonlinear and the noise terms $\mathbf{v}_{k-1}$ and $\mathbf{n}_k$ can be Gaussian or non-Gaussian. In the following section, we will try to describe a probabilistic approach to estimate the states given the observations that can handle this formulation.

2.2. Bayesian Approach

This section is written based on the paper published by Arulampalam *et al.* [8]. We try to find the state $\mathbf{x}_k$ at time *k* given the observations $\mathbf{y}_{1:k}$ by a probabilistic approach. To estimate the states, we need to construct the posterior probability density function (pdf) $p(\mathbf{x}_k \mid \mathbf{y}_{1:k})$. The estimation of the state $\mathbf{x}_k$ at time *k* can be formed in two steps:

- Prediction
- Update

2.2.1. Prediction

We want to predict the state $\mathbf{x}_k$ at time *k* from the previous states and the state noise. This stage is identical with the state model given by Eq. (2.1). If we assume that the pdf $p(\mathbf{x}_{k-1} \mid \mathbf{y}_{1:k-1})$ at time $k-1$ is available, then the prior pdf becomes

$$p(\mathbf{x}_k \mid \mathbf{y}_{1:k-1}) = \int p(\mathbf{x}_k \mid \mathbf{x}_{k-1}) p(\mathbf{x}_{k-1} \mid \mathbf{y}_{1:k-1}) d\mathbf{x}_{k-1} \tag{2.3}$$

where $p(\mathbf{x}_k \mid \mathbf{x}_{k-1})$ is the transition density.

Since the state evolution of the model is represented with the state model, the transition term $p(\mathbf{x}_k \mid \mathbf{x}_{k-1})$ in prediction stage represents the dynamics defined in Eq. (2.1). As given in a detailed justification in [9], we can show that if $\mathbf{x}_{k-1}^i$ is a sample from $p(\mathbf{x}_{k-1}, \mathbf{y}_{1:k-1})$ and $\mathbf{v}_{k-1}^i$ is a sample from $p(\mathbf{v}_{k-1})$, then $\mathbf{x}_k^i = \mathbf{f}_{k-1}(\mathbf{x}_{k-1}^i, \mathbf{v}_{k-1}^i)$ is distributed as $p(\mathbf{x}_k, \mathbf{y}_{1:k-1})$.

To implement the above equation, we assume that the initial pdf $p(\mathbf{x}_0, \mathbf{y}_0)$ is given.

2.2.2. Update

We use the new observed data to update the predicted state that is obtained in the prediction stage. Update stage is identical with the observation model given by Eq. (2.2). Now at time k, a measurement $\mathbf{y}_k$ becomes available and we can update the prior via Bayes' rule

$$p(\mathbf{x}_k \mid \mathbf{y}_{1:k}) = \frac{p(\mathbf{y}_k \mid \mathbf{x}_k)\, p(\mathbf{x}_k \mid \mathbf{y}_{1:k-1})}{p(\mathbf{y}_k \mid \mathbf{y}_{1:k-1})} \tag{2.4}$$

Here the conditional pdfs $p(\mathbf{x}_k \mid \mathbf{y}_{1:k})$, $p(\mathbf{y}_k \mid \mathbf{x}_k)$ and $p(\mathbf{x}_k \mid \mathbf{y}_{1:k-1})$ refer to the updated prior (posterior), the likelihood and the prior, respectively.

The normalizing term $p(\mathbf{y}_k \mid \mathbf{y}_{1:k-1})$ is given by the equation

$$p(\mathbf{y}_k \mid \mathbf{y}_{1:k-1}) = \int p(\mathbf{y}_k \mid \mathbf{x}_k)\, p(\mathbf{x}_k \mid \mathbf{y}_{1:k-1}) d\mathbf{x}_k \tag{2.5}$$

In the update stage, we can find the posterior density by calculating the likelihood $p(\mathbf{y}_k \mid \mathbf{x}_k)$ using the observation model in (2.2) and substituting the prior $p(\mathbf{x}_k \mid \mathbf{y}_{1:k-1})$ calculated in Eq. (2.3) into (2.4).

This posterior is used in prediction step to calculate the prior and it will again be used in update stage to find the posterior in a recursive manner. These recursive calculations make it possible to find the optimal Bayesian solution.

Since the process is iterative, it is generally difficult to find an analytic expression for the optimal solution. There are only a limited number of cases where an analytic solution exists [8]. In the following part, we will give the filter types that have an analytic formulation and in which situations they are successful. When they are insufficient, we will extend our approach with other filter types.

2.3. Kalman Filters

When the functions $\mathbf{f}_k$ and $\mathbf{h}_k$ are linear and the noise terms $\mathbf{v}_{k-1}$ and $\mathbf{n}_k$ are Gaussian, Kalman filter gives us the optimal Bayesian solution. If we write the state space formulation given in (2.1) and (2.2) in this framework, then we get the following equations

$$\mathbf{x}_k = \mathbf{F}_k \mathbf{x}_{k-1} + \mathbf{v}_{k-1} \tag{2.6}$$

$$\mathbf{y}_k = \mathbf{H}_k \mathbf{x}_k + \mathbf{n}_k \tag{2.7}$$

Here the matrices $\mathbf{F}_k$ and $\mathbf{H}_k$ define the linear functions and the noise terms, $\mathbf{v}_{k-1}$ and $\mathbf{n}_k$, are drawn from zero-mean independent Gaussian distributions $\mathcal{N}(\mathbf{v}_{k-1}; \mathbf{0}, \mathbf{Q})$, $\mathcal{N}(\mathbf{n}_k; \mathbf{0}, \mathbf{R})$ with covariances $\mathbf{Q}$ and $\mathbf{R}$, respectively.

As explained in [8], if we define the $\mathcal{N}(\mathbf{x}; \boldsymbol{\mu}, \Sigma)$ as a Gaussian density of argument $\mathbf{x}$ with mean $\boldsymbol{\mu}$ and covariance Σ, then we can give the recursive relationship of the Kalman filter as follows:

$$p(\mathbf{x}_{k-1} \mid \mathbf{y}_{1:k-1}) = \mathcal{N}(\mathbf{x}_{k-1}; \boldsymbol{\mu}_{k-1|k-1}, \Sigma_{k-1|k-1}) \tag{2.8}$$

$$p(\mathbf{x}_k \mid \mathbf{y}_{1:k-1}) = \mathcal{N}(\mathbf{x}_k; \boldsymbol{\mu}_{k|k-1}, \Sigma_{k|k-1}) \tag{2.9}$$

$$p(\mathbf{x}_k \mid \mathbf{y}_{1:k}) = \mathcal{N}(\mathbf{x}_k; \boldsymbol{\mu}_{k|k}, \Sigma_{k|k}) \tag{2.10}$$

The above equations are written to show the recursive Bayesian approach and they can be explicitly written as

$$\boldsymbol{\mu}_{k|k-1} = \mathbf{F}_k \boldsymbol{\mu}_{k-1|k-1} \tag{2.11}$$

$$\Sigma_{k|k-1} = \mathbf{Q}_{k-1} + \mathbf{F}_k \Sigma_{k-1|k-1} \mathbf{F}_k^T \tag{2.12}$$

$$\boldsymbol{\mu}_{k|k} = \boldsymbol{\mu}_{k|k-1} + \mathbf{K}_k (\mathbf{y}_k - \mathbf{H}_k \boldsymbol{\mu}_{k|k-1}) \tag{2.13}$$

$$\Sigma_{k|k} = \Sigma_{k|k-1} - \mathbf{K}_k \mathbf{H}_k \Sigma_{k|k-1} \tag{2.14}$$

where $\boldsymbol{\mu}$ is the mean, Σ is the covariance matrices, $\mathbf{F}_k$ and $\mathbf{H}_k$ are known matrices defining the linear functions and $\mathbf{K}_k$ is the Kalman gain matrix. The gain is found by

$$\mathbf{S}_k = \mathbf{H}_k \Sigma_{k|k-1} \mathbf{H}_k^T + \mathbf{R}_k \tag{2.15}$$

$$\mathbf{K}_k = \Sigma_{k|k-1} \mathbf{H}_k^T \mathbf{S}_k^{-1} \tag{2.16}$$

where $\mathbf{S}_k$ is the covariance of the error term, $\mathbf{y}_k - \mathbf{H}_k \boldsymbol{\mu}_{k|k-1}$.

The solution found by this method is the optimum solution; no algorithm can do better in this linear and Gaussian environment [8]. If the system is nonlinear, then Kalman filter fails to achieve the optimum so alternative approaches are developed to increase the performance. In the following sections, we explain extended Kalman and unscented Kalman filters that are developed to improve Kalman filter performance in those environments.

2.3.1. Extended Kalman Filters

In the previous section, we stated that the Kalman filter gives the optimum solution if the system is linear and Gaussian. When the system becomes nonlinear, Kalman filter becomes insufficient and we try to approximate the nonlinearity with local linearization of the functions. This is achieved by taking the Taylor series expansion of these nonlinear functions [10]. Extended Kalman Filter (EKF) is the nonlinear version of the Kalman filter which linearizes the nonlinear functions by taking their Taylor series expansion.

In many applications, first order Taylor series expansion can be sufficient and it becomes computationally simpler. We can find it by taking the first terms in the expansion of the nonlinear functions $\mathbf{f}_k$ and $\mathbf{h}_k$ as shown below

$$\hat{\mathbf{F}}_k = \frac{d\mathbf{f}_k(\mathbf{x})}{d\mathbf{x}} \Big|_{\mathbf{x}=\boldsymbol{\mu}_{k-1|k-1}} \tag{2.17}$$

$$\hat{\mathbf{H}}_k = \frac{d\mathbf{h}_k(\mathbf{x})}{d\mathbf{x}}\Big|_{\mathbf{x}=\boldsymbol{\mu}_{k|k-1}} \tag{2.18}$$

The recursive Bayesian approach given in Eqs. (2.8), (2.9) and (2.10) is still valid here and the mean and covariance update equations are obtained as follows:

$$\boldsymbol{\mu}_{k|k-1} = \mathbf{f}_k(\boldsymbol{\mu}_{k-1|k-1}) \tag{2.19}$$

$$\Sigma_{k|k-1} = \mathbf{Q}_{k-1} + \hat{\mathbf{F}}_k \Sigma_{k-1|k-1} \hat{\mathbf{F}}_k^T \tag{2.20}$$

$$\boldsymbol{\mu}_{k|k} = \boldsymbol{\mu}_{k|k-1} + \mathbf{K}_k(\mathbf{y}_k - \mathbf{h}_k(\boldsymbol{\mu}_{k|k-1})) \tag{2.21}$$

$$\Sigma_{k|k} = \Sigma_{k|k-1} - \mathbf{K}_k \hat{\mathbf{H}}_k \Sigma_{k|k-1} \tag{2.22}$$

where $\mathbf{K}_k$ is again the Kalman gain matrix and in this case it is computed by

$$\mathbf{S}_k = \hat{\mathbf{H}}_k \Sigma_{k|k-1} \hat{\mathbf{H}}_k^T + \mathbf{R}_k \tag{2.23}$$

$$\mathbf{K}_k = \Sigma_{k|k-1} \hat{\mathbf{H}}_k^T \mathbf{S}_k^{-1} \tag{2.24}$$

EKF gives better results than the Kalman filter if the system is nonlinear and Gaussian. Another approach, unscented Kalman filter is also developed to struggle with the system nonlinearity and it is explained in the following stage. When the data contains non-Gaussian noise, EKF can be insufficient as the Kalman filter to handle the system.

2.3.2. Unscented Kalman Filters

The Unscented Kalman Filter (UKF) is an improved version of the Kalman filter for nonlinear systems. It can give better results than EKF since it can approximate the nonlinearity more accurately. Especially we can observe its superiority when the models are highly nonlinear and the effects of higher order terms of Taylor series expansion become significant [11].

The basic idea behind this transformation is the fact that to approximate a Gaussian distribution is easier than to approximate a nonlinear function [12]. As can be seen in Fig. 2.1, the points are chosen according to a deterministic algorithm and then they are transformed to the new points applying a nonlinear function.

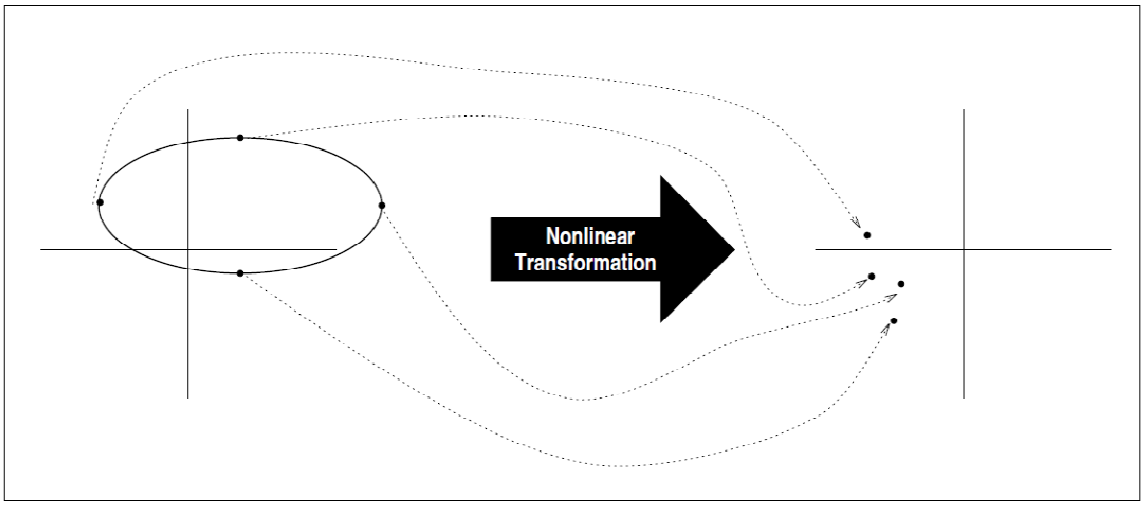


Figure 2.1. The principle of unscented transformation

2.3.2.1. The Unscented Transformation. To achieve the unscented transformation, we select a number of points to approximate a Gaussian distribution. If we assume that the mean and the covariance of the data is $\bar{\mathbf{x}}$ and $\mathbf{P}_x$, respectively; then we select these points according to a deterministic algorithm in which the mean and the covariance of the selected points are also same with the data itself. So the idea is that, we approximate a Gaussian distribution which has the same mean and variance with the data itself using these selected points.

Now we will show how these points (sigma points), $\mathbf{X}_i$'s, are selected and their weights, W_i's, are calculated. If the dimension of the data is *n*, then *2n+1* points are selected according to the equations below:

$$\mathbf{X}_0 = \bar{\mathbf{x}} \quad for\ i = 0$$
$$\mathbf{X}_i = \bar{\mathbf{x}} + (\sqrt{(n+\kappa)\mathbf{P}_x})_i \quad for\ i = 1,...,n \tag{2.25}$$

$$\mathbf{X}_i = \bar{\mathbf{x}} - (\sqrt{(n+\kappa)\mathbf{P}_x})_i \quad for\ i = n+1,...,2n$$

and the weights are found with

$$W_0 = \kappa / (n+\kappa)_i \quad for\ i = 0$$
$$W_i = \kappa / (2n+2\kappa)_i \quad for\ i = 1,...,2n \tag{2.26}$$

where κ is a scaling parameter, $(\sqrt{(n+\kappa)\mathbf{P}_x})_i$ is the *i*'th row or column of the matrix square root of $(n+\kappa)\mathbf{P}_x$ and W_i is the weight associated with the *i*'th point such that $\sum_i W_i = 1$ [11].

After the calculations are done, these sigma points are transformed to a new set of points with a nonlinear function as follows

$$\mathbf{Y}_i = \mathbf{h}(\mathbf{X}_i)\ for\ i\ =\ 1,...,2n \tag{2.27}$$

where **h** is the nonlinear function that governs the observation model equation. For these transformed points, the new mean and covariance values are found as

$$\bar{\mathbf{y}} = \sum W_i \mathbf{Y}_i \tag{2.28}$$

$$\mathbf{P}_y = \sum W_i (\mathbf{Y}_i - \bar{\mathbf{y}})(\mathbf{Y}_i - \bar{\mathbf{y}})^T \tag{2.29}$$

The new mean and covariance values are used in the implementation of UKF. It is given in Appendix A.1. UKF outperforms the EKF since the calculated values are accurate up to second order for any nonlinearity whereas EKF uses a first order Taylor series approximation for the nonlinearity.

2.4. Particle Filters

As explained in the previous sections, Kalman filter gives the optimum solution for a linear and Gaussian environment. The EKF and UKF increase the efficiency if the system contains nonlinearities. When the system is both nonlinear and non-Gaussian, then particle filters yield an improvement in performance.

In particle filter approach, the basic idea is to approximate the posterior density with a set of random samples. As the number of samples increases, the distribution is represented more accurately and the state estimates can be obtained from these samples [8].

To understand the process more clearly, it would be useful to explain the Monte Carlo methods [10] since particle filtering itself is a Sequential Monte Carlo approach [13].

2.4.1. Monte Carlo Methods

Monte Carlo methods are widely used in physical and mathematical systems and they rely on taking random samples to represent the true dynamics. Especially in mathematics, sometimes it becomes very hard to take the integral of a function analytically and in those circumstances, Monte Carlo integration is used to randomly select some samples and evaluate the integral using them.

Suppose that we want to evaluate the integral

$$I = \int p(x)dx \tag{2.30}$$

If we can draw samples from a known distribution $q(x)$ that is close to the target distribution $p(x)$, then we can find the sample mean as

$$I_N = \frac{1}{N}\sum_{i=1}^{N} q(x^i)w(x^i) \tag{2.31}$$

where $\{x^i; i = 1,..., N\}$ are the samples drawn from the known distribution, $w(x^i)$ and $q(x^i)$ are the weights and the values of the samples, respectively. The distribution q(x) is also known as importance or proposal distribution. We normalize the weights so that the equation $\sum_i w(x^i) = 1$ should be ensured [10].

2.4.2. Sequential Importance Sampling

Sequential importance sampling (SIS) is a reference technique for most sequential Monte Carlo methods. In this technique, to approximate the true posterior density, again state and observation models will be used as we discussed in section 2.1.

Now, from a Bayesian perspective suppose that at the k'th iteration we approximate the posterior pdf $p(\mathbf{x}_{0:k} \mid \mathbf{y}_{1:k})$ with the sample points $\{\mathbf{x}_{0:k}^i; i = 0,..., N\}$ and their associated weights $\{w_k^i; i = 1,..., N\}$.

Then, similar to Eq. (2.31), the posterior density at time k can be approximated as

$$p(\mathbf{x}_{0:k} \mid \mathbf{y}_{1:k}) \approx \sum_{i=1}^{N} w_k^i \delta(\mathbf{x}_{0:k} - \mathbf{x}_{0:k}^i) \tag{2.32}$$

Here the weights are chosen using the importance sampling which is discussed in detail in [8], because some of the particles in the distribution have more effect than the others so their importance should be emphasized. If the samples $\mathbf{x}_{0:k}^i$ were drawn from an importance density $q(\mathbf{x}_{0:k} \mid \mathbf{y}_{1:k})$, then the weights in (2.32) becomes

$$w_k^i \propto \frac{p(\mathbf{x}_{0:k}^i \mid \mathbf{y}_{1:k})}{q(\mathbf{x}_{0:k}^i \mid \mathbf{y}_{1:k})} \quad for\ i = 1,..., N \tag{2.33}$$

As the new observations become available, we can approximate $p(\mathbf{x}_{0:k} \mid \mathbf{y}_{1:k})$ using the samples which approximate the distribution of $p(\mathbf{x}_{0:k-1} \mid \mathbf{y}_{1:k-1})$. Then the weight update equation can be shown to be

$$w_k^i \propto w_{k-1}^i \frac{p(\mathbf{y}_k \mid \mathbf{x}_k^i)\, p(\mathbf{x}_k^i \mid \mathbf{x}_{k-1}^i)}{q(\mathbf{x}_k^i \mid \mathbf{x}_{0:k-1}^i, \mathbf{y}_{1:k})} \quad for\ i = 1,\ldots,N \tag{2.34}$$

The history of observations and states are not needed to be stored if we assume that $q(\mathbf{x}_k \mid \mathbf{x}_{0:k-1}, \mathbf{y}_{1:k}) = q(\mathbf{x}_k \mid \mathbf{x}_{k-1}, \mathbf{y}_k)$. Then the weight update equation in (2.34) becomes

$$w_k^i \propto w_{k-1}^i \frac{p(\mathbf{y}_k \mid \mathbf{x}_k^i)\, p(\mathbf{x}_k^i \mid \mathbf{x}_{k-1}^i)}{q(\mathbf{x}_k^i \mid \mathbf{x}_{k-1}^i, \mathbf{y}_k)} \quad for\ i = 1,\ldots,N \tag{2.35}$$

and the posterior in (2.32) now can be approximated as

$$p(\mathbf{x}_k \mid \mathbf{y}_{1:k}) \approx \sum_{i=1}^{N} w_k^i \delta(\mathbf{x}_k - \mathbf{x}_k^i) \tag{2.36}$$

Thus as each observation becomes available, the importance weights w_k^i and the sample points $\mathbf{x}_k^i$ are propagated iteratively. SIS algorithm in given in Appendix A.2 as a closed form.

One of the drawbacks in particle filter approach is the degeneracy problem [10]. Degeneracy means that after a few iterations, most of the particles will have negligible weights except one particle. It decreases the diversity so it is perceived as an undesirable property. It can be solved by using a very large number of samples but it is often impractical so two other methods are proposed:

- good choice of importance density
- use of resampling

2.4.2.1. Good Choice of Importance Density. There have been considerable research done to choose the importance density properly [10]. To choose the importance density to be the transition prior is often convenient and practical:

$$q(\mathbf{x}_k \mid \mathbf{x}_{k-1}^i, \mathbf{y}_k) = p(\mathbf{x}_k \mid \mathbf{x}_{k-1}^i) \tag{2.37}$$

Then the weight update equation in (2.35) becomes

$$w_k^i \propto w_{k-1}^i p(\mathbf{y}_k \mid \mathbf{x}_k^i) \tag{2.38}$$

Although transition prior is practical to take as importance density, it has some drawbacks. If it is much broader than the likelihood, then only a few particles will be assigned a high weight and it would again cause degeneracy problem.

To choose the particles in the right region where the high likelihood exists, there have been some methods proposed. Making the distribution coincident through the prior and the likelihood [14] or using local linearization techniques to approximate the optimal importance density [11] are two of the methods researched in the literature.

2.4.2.2. Resampling. The basic idea of resampling is to eliminate the particles that have small weights and to concentrate on particles with large weights. After resampling, a new set $\{\mathbf{x}_k^i\}_{i=1}^N$ is generated and their weights are now reset to $w_k^i = 1/N$. Resampling step [13] is shown in Fig. 2.2:

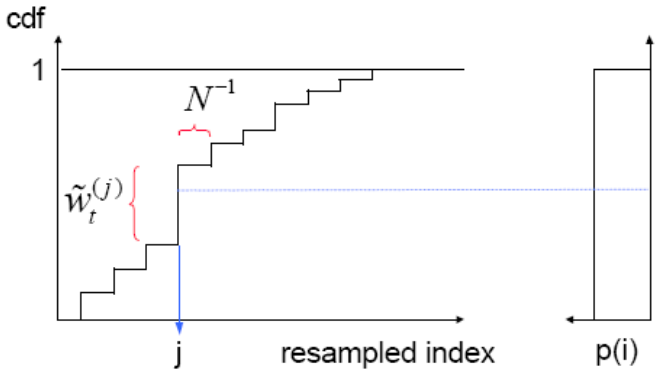


Figure 2.2. The process of resampling

In resampling technique, we first form the cumulative distribution function (cdf) of the weights and then select a number randomly as shown in Fig. 2.2. The index of the weight forming the cdf that coincides with the random number determines the new selected particle. As the weight of the particle increases, it is more probable to be selected.

Although the resampling step reduces the effects of the degeneracy problem, it introduces other practical problems [8]. First, it limits the opportunity to parallelize since all the particles must be combined. Second, the particles that have high weights are statistically selected many times and this leads to a loss of diversity among the particles. This problem, which is known as *sample impoverishment*, is severe in the case of small process noise.

There have been some systematic techniques proposed recently to solve this problem such as resample-move [15] or regularization [8] which are discussed in given resources.

The resampling algorithm and sampling importance resampling (SIR) algorithm is given in Appendix A.3 and A.4. A single cycle of the particle filter with resampling step is shown in Fig. 2.3 [16]:

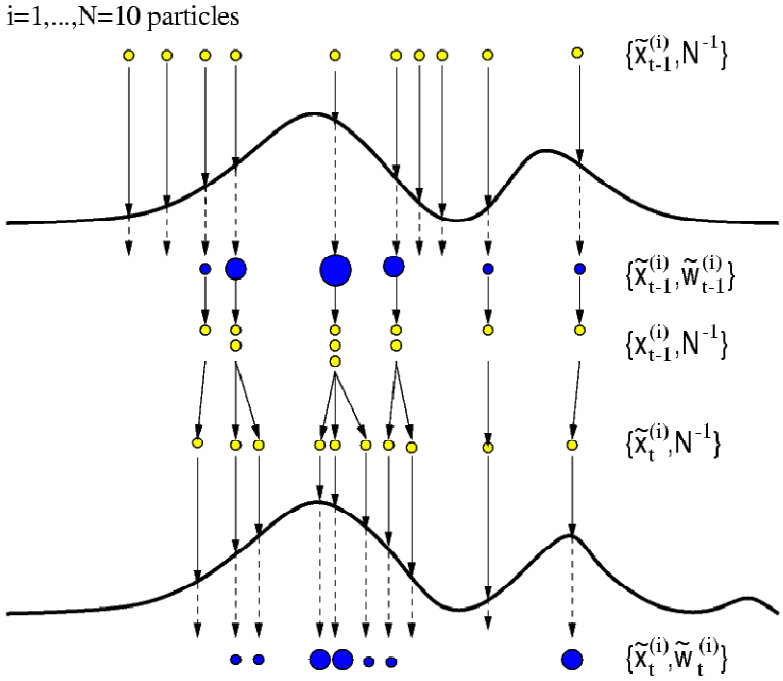


Figure 2.3. A single cyle of particle filter

2.4.3. Auxiliary Particle Filter

The original auxilliary sampling importance resampling (ASIR) filter consists of one more step compared to particle filter, namely, a resampling stage to produce an i.i.d sample $\{\mathbf{x}_k^j, i^j\}_{j=1}^M$ with equal weights. Compared with the SIR filter, the advantage of the ASIR filter is that it naturally generates points from the sample at time $k-1$, and conditioned on the current measurement, they are most likely to be close to the true state. It gives better results if the likelihood is situated in the tails of the prior [11] as shown in Fig. 2.4.

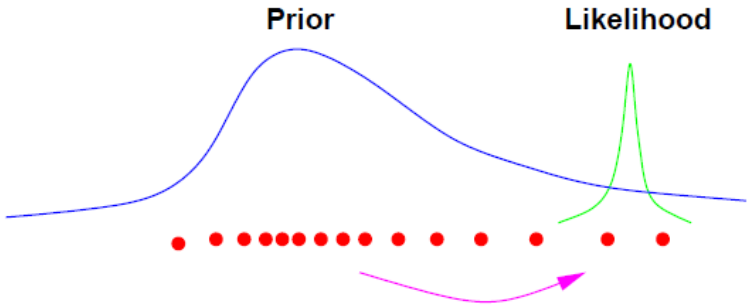


Figure 2.4. If the likelihood happens to lie in one of the tails of the prior distribution, ASIR can give better results than the particle filter.

ASIR can be viewed as resampling at the previous time step, based on some point estimates τ_k^i that characterize $p(\mathbf{x}_k \mid \mathbf{x}_{k-1}^i)$. If the process noise is small so that $p(\mathbf{x}_k \mid \mathbf{x}_{k-1}^i)$ is well characterized by τ_k^i, then ASIR is often not so sensitive to outliers as SIR, and the weights w_k^i are more even. However, if the process noise is large, a single point does not characterize $p(\mathbf{x}_k \mid \mathbf{x}_{k-1}^i)$ well, and ASIR resamples based on a poor approximation of $p(\mathbf{x}_k \mid \mathbf{x}_{k-1}^i)$. In such scenarios, the use of ASIR then degrades performance [8].

The algorithm for the ASIR filter is given in Appendix A.5.

2.5. Texture Models

By a model of texture, we mean a mathematical process which creates or describes the textured image. The goal of texture modelling is the description and the classification of the texture images.

We can find several texture models appeared in the literature but for our case, we will focus on linear and Markov Random Field (MRF) models.

2.5.1. One Dimensional Linear Texture Models

In this model type, the prediction of the future data is made on the previous variable sequence like in time series. If the image is converted to row vectors (snake form) as shown in Fig. 2.4 [7], then we can formulate the relationship between the pixels as follows:

$$x(t) = \sum_{n} a_n x(t-n) + v(t) \tag{2.39}$$

where $x(t)$ is the gray level at point t, $a_n's$ are the one dimensional autoregressive (1-D AR) coefficients and the $v(t)$ is the i.i.d noise variable.

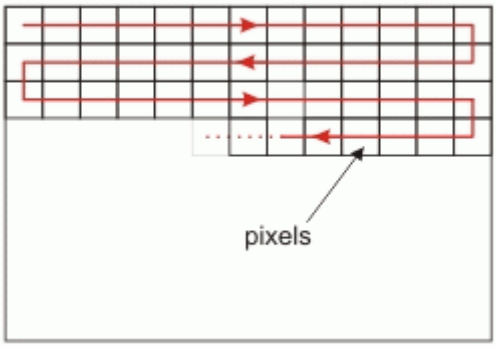


Figure 2.5. Snake form representation of the image

2.5.2. Markov Random Field Models

The use of the neighbors offers a geometric framework for the clustering of pixel intensities and the Markov property is a natural way to formalize this notion [17]. In this model type, the brightness level at a point in the image is highly dependent on the

brightness levels of neighboring points. While describing the model, the number of neighborhood points that are used to assign the target point is related with the model order. For instance, in Fig. 2.5 a first order MRF model is shown and in this model order, the center pixel value *C* is dependent on its four neighbors: *u*, *u'*, *t*, *t'*. The conditional probability does take the form

$$p(C = g \mid u, u', t, t') = \frac{\exp(gT)}{1 + \exp(T)} \tag{2.40}$$

where

$$T = a + b_1(t + t') + b_2(u + u') \tag{2.41}$$

for the binomial case. Here, a, b_1 and b_2 are the MRF parameters that control the clustering in the texture. For the binomial case, the data can take the values just only zero or one.

If the gray level range changes between *0* and *G-1*, then the conditional probability takes the form [18]:

$$p(C = g \mid T) = \binom{n}{g} \psi^g (1 + \psi)^{-n} \tag{2.42}$$

where $\psi = \exp(T)$ and *n* is the maximum gray level. Here *g* can take the values between *0* and *G-1*.

	u'	
t	**C**	t'
	u	

Figure 2.6. First order MRF model

In MRF model, the technique used to estimate the parameters is Maximum Likelihood Estimation (MLE). The usual likelihood, *l*, is given by

$$l = \sum_{\text{Neighbors of X}} \ln(p(X \mid .)) \tag{2.43}$$

where $p(X \mid .)$ denote the conditional probability $p(X = x \mid Neighbors\ of\ X)$. To make the calculation simpler, the lattice is partitioned to the codings [19], which mean disjoint sets of points.

Then $l(i)$ refers to the log likelihood of *i*'th coding and found with

$$l(i) = \sum_{\substack{\text{Support points} \\ \text{of the coding}}} \ln(p(X \mid .)) \tag{2.44}$$

We try to maximize $l(i)$ to find the parameters a and b_j so that

$$\partial l(i) \Big/ \partial a = 0 \quad i = 1,...,Number\ of\ codings \tag{2.45}$$

$$\partial l(i) \Big/ \partial b_j = 0 \quad \begin{matrix} i = 1,...,Number\ of\ codings \\ j = 1,...,Euclidean\ neighborhood\ number \end{matrix} \tag{2.46}$$

Using Newton's method, we solve the equation and find the estimated parameter values for $\{ a, b_j \}$.

2.5.3. Two Dimensional Linear Texture Models

In this model type, we do not transform the image into a row vector; in contrast we try to find a dependency between the image pixel values in a two dimensional (2-D) space. The brightness level at point *(i, j)* is found by the following equation

$$x(i, j) = \sum_m \sum_n a_{m,n} x(i - m, j - n) + v(i, j) \tag{2.47}$$

where $a_{m,n}'s$ are the 2-D AR coefficients and *v(i, j)* is the i.i.d noise variable.

In Fig. 2.6, we can see 2-D linear texture model with a 3x3 prediction mask and in this prediction mask size, the pixel value at point (i,j) is dependent on the previous eight points. There is no restriction on prediction mask size to specify but it would be practical to choose it as small as possible for computational simplicity.

i-2, j-2	i-2, j-1	i-2, j
i-1, j-2	i-1, j-1	i-1, j
i, j-2	i, j-1	**i, j**

Figure 2.7. The brightness level at point (i, j) is dependent on neighbor pixels

2.6. Noise Descriptions

In filtering process, determining the correct noise type is as crucial as choosing the correct texture model. Some of the noise descriptions in the literature are investigated and checked how suitable they are for the textile fabric images.

2.6.1. Gaussian Noise

It is one of the most important noise types that are used in many signal processing applications. It is a statistical noise that has a probability density function of the normal distribution. The parameters mean and variance determines the distribution and the values that the noise can take are chosen from it.

Gauss distributions have the pdf $\frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(x-\mu)^2}{2\sigma^2}}$ and they have steeper slopes as the variance of the distribution decreases. With different parameters sets several plots can be seen in Fig. 2.7 [20].

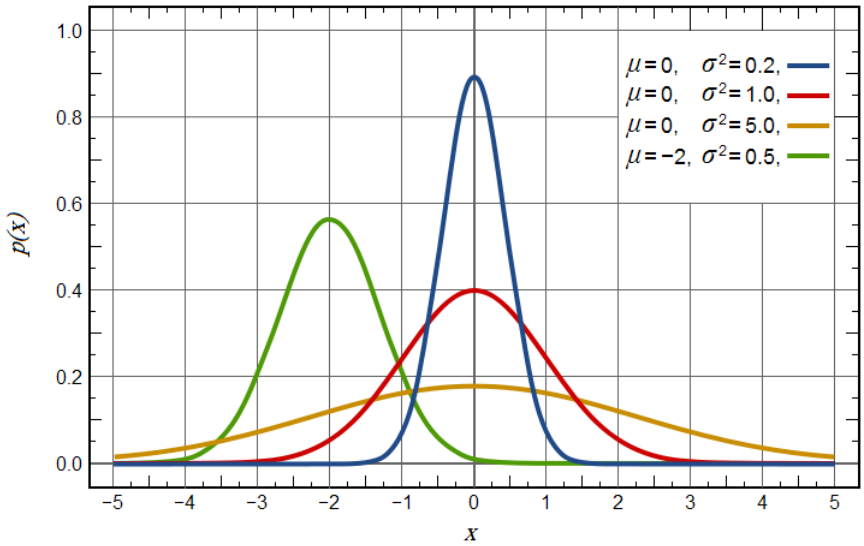


Figure 2.8. Normal distributions with different parameter sets

2.6.2. Alpha Stable Noise (αS Noise)

This noise type can show great success if the data is too impulsive where the Gaussian noise can be insufficient to model it [21]. The αS distribution family is described most conveniently by its characteristic function, $\varphi(t)$, which is the Fourier transform of the pdf:

$$\varphi(t)=\exp\left\{j\delta t-\gamma|t|^{\alpha}\left[1+j\beta\,\mathrm{sgn}(t)\tan\left(\frac{\alpha\pi}{2}\right)\right]\right\}\quad if\quad \alpha\neq 1$$
$$\varphi(t)=\exp\left\{j\delta t-\gamma|t|^{\alpha}\left[1+j\beta\,\mathrm{sgn}(t)\frac{2}{\pi}\log|t|\right]\right\}\quad if\quad \alpha=1 \tag{2.48}$$

where $\alpha\in(0,2]$ is the *characteristic exponent* which sets the impulsiveness of the distribution, $\beta\in[-1,1]$ is the *symmetry parameter* which sets the skewness, γ > 0 (*dispersion*) is the scale parameter and is analogous to the variance and $\delta\in(-\infty,\infty)$ is the location parameter which represents the shift from the origin [21].

When $\beta=0$, the distribution is symmetric around μ and it is called a *symmetric α-stable (SαS) distribution*. When $\delta=0$, the distribution is centralized at the origin and when ($\alpha=2$), we have a compact density function which is a Gaussian distribution.

To give a better understanding of the behaviour of these distributions, we provide plots for various parameter values in Figs. 2.8 and 2.9. From the plots, we can observe that as the characteristic exponent decreases, the slope becomes steeper and the tails of the distribution gets heavier.

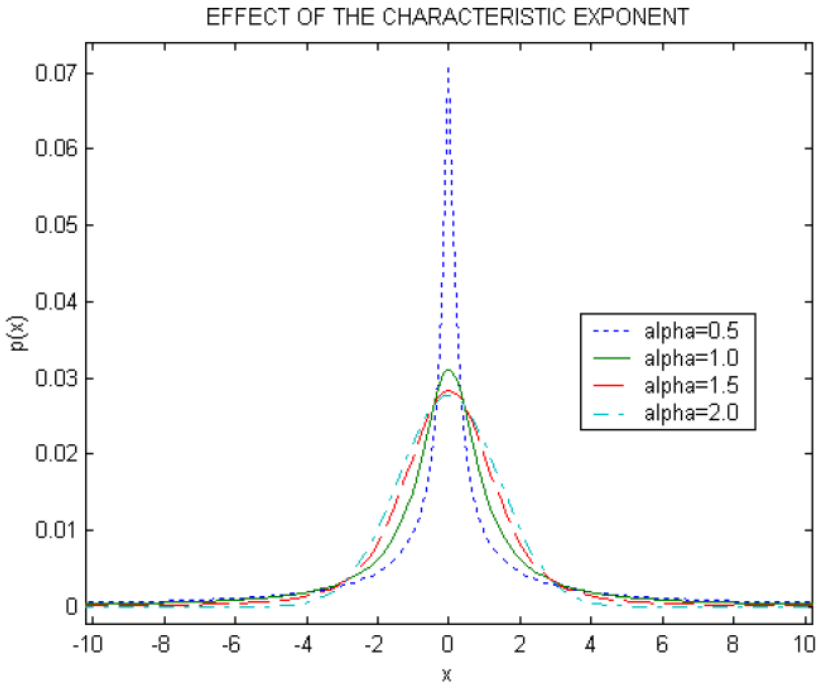


Figure 2.9. Effect of the characteristic exponent on the general distribution

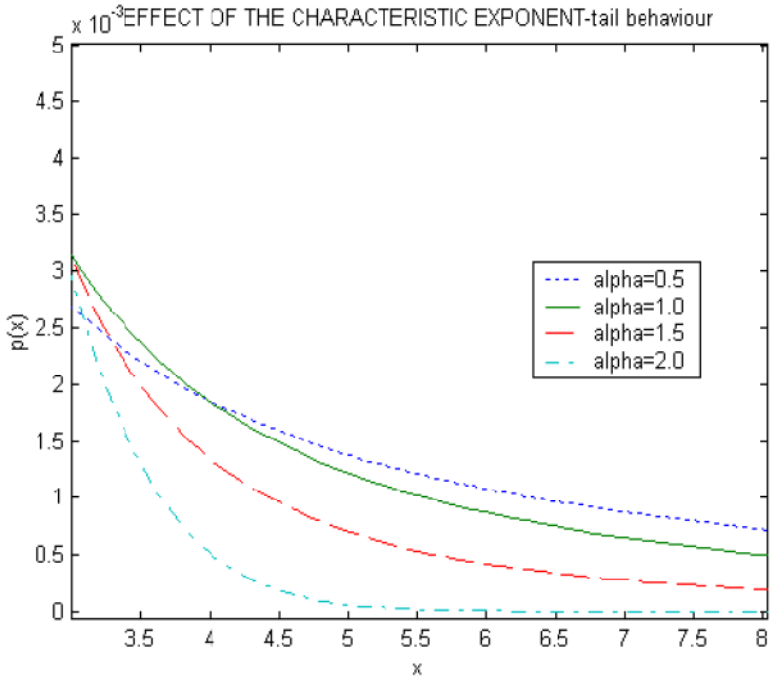


Figure 2.10. Effect of the characteristic exponent on the tails

2.6.3. Mixture of Gaussian Noise

In this noise type, the probability distribution is a combination of several Gaussian distributions. The new Mixture of Gaussian (MoG) distribution will have the pdf *p(x)* as

$$p(x) = \sum_{i=1}^{n_c} c_i \mathcal{N}_i(z_i, \mu_i, \sigma_i^2) \tag{2.49}$$

where n_c is the number of clusters, c_i is the mixture proportion and $\mathcal{N}_i(z_i, \mu_i, \sigma_i^2)$ is the contributing distribution to the mixture of argument z_i with mean μ_i and variance σ_i^2. Fig. 2.10 shows that just three $\mathcal{N}(x, \mu, \sigma^2)$ Gaussian distributions can model a complex, multi-modal distribution [22].

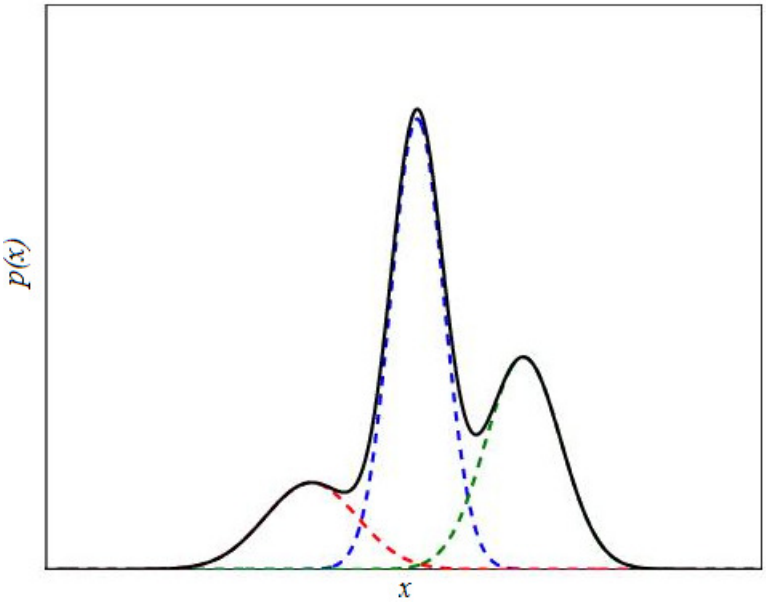


Figure 2.11. The construction of MoG distribution with local Gaussians

3. IMPLEMENTATIONS AND RESULTS

In this section, we begin with describing the experimental setup that is used in the implementations and the methodology that we have followed. Then, based on the state space model, the system description and the parameter estimation methods are explained. Finally, we give the filter implementations, their results and a comparative time performance analysis.

3.1. Experimental Setup and Methodology

The textile fabric images that we have used in all our implementations are provided from the TILDA database [23]. In this database, there are several types of textile fabrics and for each textile fabric type, there exist a defect-free image folder and eight different defective image folders. In the TILDA database, the image folders ending with "EO" represent the defect free image folders. Although the defective images are grouped in eight different folders; while running the algorithms, we implemented only four of the them since the defects were evident only in those folders. Each folder includes 50 images so we had 50 defect-free and 200 defective images totally for a given textile type.

The methodology we have followed for finding the defects has two steps, namely, learning stage and defect detection stage.

In learning stage, mainly two important things are done. One of them is to calculate the 1-D or 2-D AR coefficients of a given textile type to determine the state model defined in (2.1) and the other thing is to find the true state vector of the textile fabrics that represents the texture. To accomplish these tasks, each defect free image of a given textile type is divided into nonoverlapping sub-windows and the AR parameters depending on the model order are calculated for each sub-window. Then, computing the mean of these values, the average AR parameters modeling the given textile type are found.

After calculating the AR parameters, their values are substituted into the state model equation and the hidden states are estimated using the state and observation models

for each sub-window of the defect free images. Then the average of these vectors are taken in order to average out intensity changes due to noise and camera effects and the true state vector is found that represents the texture. Calculation of the AR parameters and the estimation of the true state vectors are done offline only once for each textile type.

In the defect detection stage, the test image is again divided into nonoverlapping sub-windows. Then for each sub-window, we substitute the AR parameters obtained in the learning stage into the state equation and find the corresponding state vectors. Then, comparing these vectors using a distance measure with the true state vector, we label the sub-window as defective if the distance between the vectors are above a threshold value.

The distance measure used between the state vectors is Euclidean distance. For *i*'th sub-window, it is found by $d_i = \sqrt{\sum_{j=1}^{SL} (\mathbf{x}_{true}^{j} - \mathbf{x}_{test}^{j})^2}$ where *SL* is the length of the state vectors. The sub-window is labelled as defective if d_i is above a threshold value. This threshold value is calculated by $Th = D_m + \lambda(D_u - D_l)$ where $\boldsymbol{D}$ is the distance vector consisting of d_i's; D_m, D_u and D_l are the median of $\boldsymbol{D}$, median of the distances bigger than D_m and median of distances smaller than D_m, respectively [7]. λ is some scaling parameter and it can be found by trial and error. In our filter implementations, it was taken as 2.55.

The evaluation of the performance of the filters is a little complicated. Since the defect locations are not given explicitly, we used the image folders that end with "E0" to calculate the false alarm rate. We assume that all the image surfaces are free of the defects in that folder and counting the number of windows as labeled defective gives us the false alarm rate. On the other hand, to measure the correct detection rate, we investigate the test images and if the defect is labelled correctly, then we ignore the false alarms on it and assume that image as correctly classified.

The implementation of the filters were performed with different sub-window sizes. As discussed in [7], the usage of 32x32 sub-window blocks were practical and convenient so we compared all the filter results in this manner. The resolution of the images we tested

was 512x768 and dividing each image into sub-windows, we obtained 384 sub-windows for each image. In the defect free image folder, there were 50 images so we had 19200 sub-windows totally.

In the following sections, we will discuss the approaches that we used to find the defects.

3.2. System Description

As discussed in the theoretical background in section 2.1, we used a state space approach for modeling the textile fabric images and finding the defects. The state and observation models are given by Eqs. (2.1) and (2.2), respectively.

The cases based on the state model that are analyzed are listed below:

- The state model is 1-D linear and the noise is Gaussian
- The state model is 1-D linear and the noise is non-Gaussian
- The state model is 2-D linear and the noise is Gaussian
- The state model is 2-D linear and the noise is non-Gaussian

Since we assume that the data we investigate are noisy measurements, we admit the observation model as linear and we analyze the following cases for the observation model:

- The observation model is linear and the noise is Gaussian
- The observation model is linear and the noise is non-Gaussian

In Table 3.1, we give the possible options that we will discuss in this section.

For the first case of Table 3.1, we will use the following equations to model the state space dynamics as linear and Gaussian.

$$\mathbf{x}(t) = a_1\mathbf{x}(t-1) + a_2\mathbf{x}(t-2) + \mathbf{v}(t) \tag{3.1}$$

$$\mathbf{y}(t) = \mathbf{x}(t) + \mathbf{n}(t) \tag{3.2}$$

Here, a_1 and a_2 represent the 1-D AR coefficients and **v**(t) and **n**(t) the measurement and the observations noise terms, respectively. The noise terms are Gaussian.

Table 3.1. State and observation model combinations

State / Observation Model	**Linear, Gaussian**	**Linear, non-Gaussian**
1-D Linear, Gaussian	Case 1	Case 2
1-D Linear, non-Gaussian	Case 3	Case 4
2-D Linear, Gaussian	Case 5	Case 6
2-D Linear, non-Gaussian	Case 7	Case 8

For Case 4, again the Eqs. (3.1) and (3.2) are valid but now the noise terms are non-Gaussian. The non-Gaussian noise descriptions we investigated are:

- αS noise
- MoG noise

In the cases five to eight, we use 2-D linear texture models in the state model equation. We investigated the MRF model if it can be used as a texture model instead of 1-D linear model however they are not appropriate for filtering purposes due to their non-causality.

So we tried to make the model similar to the MRF model that is shown in Fig. 3.1 and we approximated the system with a 2-D linear model as shown in Fig. 3.2.

In the MRF model, the center pixel is dependent on the surrounding neighbor pixels. Since the forward pixels are important, we cannot use it for filtering so we make the system casual by shifting the target pixel to the corner as shown in Fig. 3.2. This way our model becomes a 2-D casual linear model.

w	u’	v’
t	**C**	t’
v	u	w’

Figure 3.1. Second order MRF model

The state model representation of this model that is defined in Fig. 3.2 can be given explicitly as follows:

$$x(i, j) = a_{0,1}x(i, j-1) + a_{0,2}x(i, j-2) + a_{1,0}x(i-1, j) + a_{1,1}x(i-1, j-1) + ...$$
$$...a_{1,2}x(i-1, j-2) + a_{2,0}x(i-2, j) + a_{2,1}x(i-2, j-1) + a_{2,2}x(i-2, j-2) + v(i, j)$$

w	u’	v’
t	w’	t’
v	u	**C**

Figure 3.2. Two dimensional linear model

In implementation section, for the linear case we will analyze 1-D linear model and for the noncasual case, we will replace it with 2-D casual linear model.

In the next section, we will explain the estimation procedure for 1-D and 2-D AR coefficients that are used in the state model.

3.3. Estimation of One Dimensional AR Coefficients

In the state equation of the linear state model, the parameters a_1 and a_2 represent the AR coefficients and **v**(t) the noise. If the noise term is Gaussian, then we can estimate the AR parameters with the least squares (LS) method. However noise can be non-Gaussian and in that case, LS approach can diverge from the true values. In those cases when the system includes outliers, robust ridge regression methods are more successful.

The model dynamics is represented with Eq. (3.1). LS estimation will be used for Case 1, Case 2 and iteratively reweighted least squares (IRLS) estimation, which is a robust ridge regression method, will be used for Case 3 and Case 4.

3.3.1. Least Squares Estimation

The most common representation for the linear prediction is

$$\hat{x}(t) = -\sum_{n=1}^{p} a_n x(t-n) \tag{3.3}$$

where $\hat{x}(t)$ is the predicted signal value, *x(t − n)*'s are the previous observed values, and a_n's are the predictor coefficients.

We would like to minimize the error in the predicted value and the actual value, namely, $\mathbf{x}_i - \hat{\mathbf{x}}_i$ in the LS sense. This can be formulated by minimizing the sum of the square of the magnitude of the error vector [24]

$$\sum_{i=1}^{n} \left| \mathbf{x}_i - \hat{\mathbf{x}}_i \right|^2 \tag{3.4}$$

Minimizing (3.4) leads to the equation

$$\hat{\mathbf{a}} = (\mathbf{X}^T \mathbf{X})^{-1} \mathbf{X}^T \mathbf{Y} \tag{3.5}$$

where **Y** is a vector of response variables, **X** is a matrix of predictor variables, namely the past values and **â** is the vector of 1-D AR coefficients. They are defined as follows

$$\hat{\mathbf{a}} = \begin{bmatrix} a_1 & a_2 & \cdots & a_p \end{bmatrix}^T \tag{3.6}$$

$$\mathbf{X} = \begin{bmatrix} x(t-p) & x(t-p+1) & \cdots & x(t-1) \\ x(t-p-1) & x(t-p) & \cdots & x(t-2) \\ \vdots & \vdots & \ddots & \vdots \\ x(t-T+1) & x(t-T+2) & \cdots & x(t-T+p) \end{bmatrix} \tag{3.7}$$

$$\mathbf{Y} = \begin{bmatrix} x(t-p-1) & x(t-p-2) & \cdots & x(t-T) \end{bmatrix}^T \tag{3.8}$$

where p is the order of the model [25].

3.3.2. Iteratively Reweighted Least Squares Estimation

When there are outliers or collinearity in the observations, the LS approach can be unreliable. In those cases, robust ridge estimator is a popular alternative to LS and the new goal becomes to minimize

$$\sum_{i=1}^{n} \rho(\mathbf{x}_i - \hat{\mathbf{x}}_i) \tag{3.9}$$

for some function ρ. We call ρ as the objective function and in [26], various functions are given that would be suitable for this purpose. In the LS case, we can see that the minimization becomes equivalent to (3.4). In Fig. 3.3, we display the objective function $\rho(x) = x^2$ in comparison to $\rho(x) = \log(1 + x^2)$ for scalars.

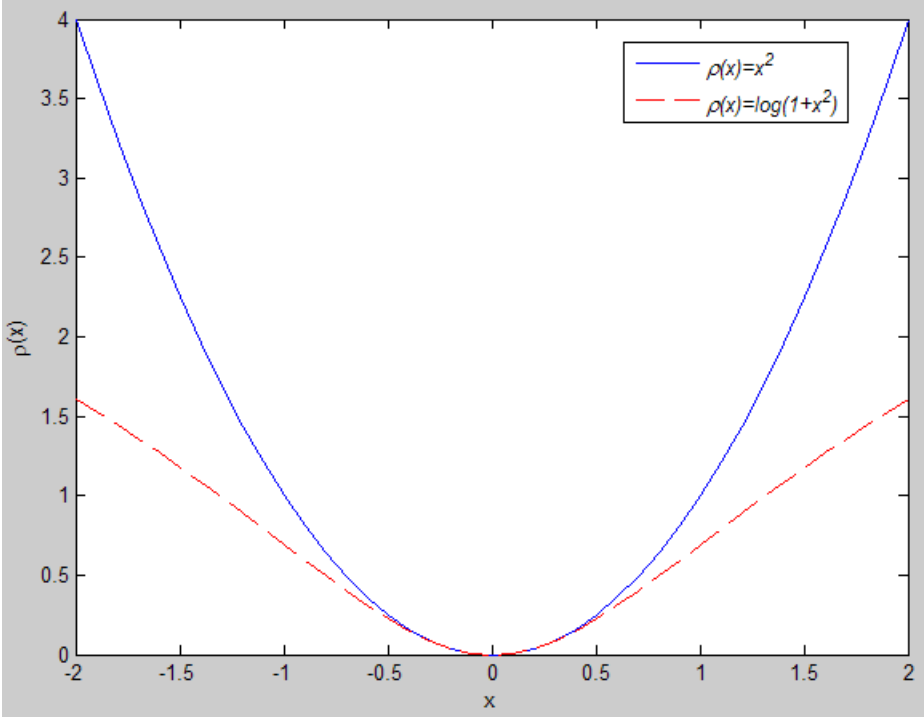


Figure 3.3. Plot of the objective functions

Combining ridge and robust estimator, the solution for the 1-D AR parameters is obtained as

$$\hat{\mathbf{a}} = (\mathbf{X}^T \mathbf{\Lambda} \mathbf{X} + \eta \mathbf{I})^{-1} \mathbf{X}^T \mathbf{\Lambda} \mathbf{Y} \tag{3.10}$$

where η is the ridge parameter and $\mathbf{\Lambda}$ is the diagonal elements with

$$\zeta_i = \frac{\rho'((\mathbf{x} - \hat{\mathbf{x}})/\theta)}{2(\mathbf{x} - \hat{\mathbf{x}})/\theta} \tag{3.11}$$

Here θ is a robust scale parameter [26].

When noise is Gaussian, the solution given in (3.10) reduces to LS solution since the matrix consisting of diagonal elements, $\mathbf{\Lambda}$, evolves to an identity matrix.

3.3.3. Comparison of the Estimation Algorithms

In this section, we compare the accuracy of the LS and IRLS algorithms both in generated synthetic images and the textile fabric images. We take the 1-D linear texture model order, *p*, as two because it lessens the computational burden [7].

In Table 3.2, you can see the 1-D AR coefficients estimated by LS and IRLS methods. In IRLS method, several objective functions are tried defined in Eq. (3.9) and their names are given near the method name in the table. We take the ridge parameter η as zero.

The estimation procedure is accomplished with first generating the synthetic images with known AR parameters and non-Gaussian noise. We preferred the noise term as non-Gaussian because it would be easier to see the differences between these two algorithms. The AR parameters $a_1 = 1.24$ and $a_2 = -0.48$ are used to generate the images. As a non-Gaussian noise term, MoG is formed with three clusters with means -9, 7, 2 and variances 20, 12 and 4, respectively.

Table 3.2. Estimated AR coefficients by using LS and IRLS methods. The names in the parenthesis near the IRLS method are, different objective function calls. The image is generated with $a_1 = 1.24$ and $a_2 = -0.48$. MoG noise is used consisting of three clusters with means -9, 7, 2 and variances 20, 12 and 4, respectively.

Methods	**Estimated** a_1	**Estimated** a_2
IRLS('andrews')	1.1761	-0.4229
IRLS('bisquare')	1.1758	-0.4227
IRLS('cauchy')	1.1658	-0.4164
IRLS('fair')	1.1583	-0.412
IRLS('huber')	1.1644	-0.4158
LS	1.1295	-0.3929

In Table 3.3, we give the estimated 1-D AR coefficients obtained from the textile fabric images. To get these parameters, the defect free images are used and after calculating the values for each image, their average is taken.

Table 3.3. Estimated AR coefficients from the textile fabric images

Methods	**Estimated** a_1	**Estimated** a_2
IRLS('andrews')	1.2311	-0.4887
IRLS('bisquare')	1.231	-0.4886
IRLS('cauchy')	1.2285	-0.4866
IRLS('fair')	1.2266	-0.4848
IRLS('huber')	1.2281	-0.4858
LS	1.0951	-0.0966

3.4. Estimation of Two Dimensional AR Coefficients

In this stage, we find the 2-D AR coefficients using LS of 2-D prediction error so it would be thought that the noise is Gaussian. We can represent our system with Eq. (2.47) and find the residual error as

$$E = \sum_i \sum_j e^2(i, j) \tag{3.12}$$

where, e is the error for each estimated data and it is defined as

$$e(i, j) = x(i, j) - \sum_m \sum_n a_{m,n} x(i - m, j - n) \tag{3.13}$$

To minimize the residual error in (3.12), we solve the equation

$$\mathbf{R\hat{a}} = \mathbf{r} \tag{3.14}$$

where the parameters **â, r** and **R** are defined as follows [27]:

$$\hat{\mathbf{a}} = \begin{bmatrix} a_{0,1} & a_{0,2} & \cdots & a_{S-1,S-1} \end{bmatrix}^T \tag{3.15}$$

$$\mathbf{r} = \begin{bmatrix} \phi(0:1) & \phi(0:2) & \cdots & \phi(0:p) \end{bmatrix}^T \tag{3.16}$$

$$\mathbf{R} = \begin{bmatrix} \phi(1:1) & \phi(2:1) & \cdots & \phi(p:1) \\ \phi(1:2) & \phi(2:2) & \cdots & \phi(p:1) \\ \vdots & \vdots & \ddots & \vdots \\ \phi(1:p) & \phi(2:P) & \cdots & \phi(p:p) \end{bmatrix} \tag{3.17}$$

Here, **â** is the 2-D AR coefficients vector, **r** is the correlation vector and **R** is the correlation lag matrix. The number of prediciton coefficients *p* ise determined according to the prediction mask size. If the mask size is defined with a *S*x*S* quarter plane, then *p* is found as $p = S^2 - 1$.

The elements $\phi(d_1, d_2)$'s in the correlation vector **r** and the correlation matrix **R** are actually one dimensional mapping of the 2-D correlation entries.

$$\phi(d_1, d_2) = \sum_m \sum_n x(m - \frac{d_1}{S}, n - \mathrm{mod}(d_1, S)) x(m - \frac{d_2}{S}, n - \mathrm{mod}(d_2, S)) \tag{3.18}$$

The solution procedure after this step is similar to the one explained in LS. Applying the algorithm to textile fabric images with a 3x3 prediction mask, we get the estimated 2-D AR coefficients as follows:

$$\hat{\mathbf{a}} = \begin{bmatrix} a_{2,2} & a_{2,1} & a_{2,0} \\ a_{1,2} & a_{1,1} & a_{1,0} \\ a_{0,2} & a_{0,1} & 1.00 \end{bmatrix} = \begin{bmatrix} 0.12 & 0.03 & -0.44 \\ -0.08 & -0.47 & 1.26 \\ -0.02 & 0.60 & 1.00 \end{bmatrix}$$

So far we have tried to find the AR parameters of the state model for different cases. Now we can look at each of these cases that are cited in Table 3.1.

3.5. Case Implementations and Results

In this section, the state and observation model combinations given in Table 3.1 are going to be implemented one by one. To compare the performances of our Bayesian approach and the filter types, we give the results obtained from another texture analysis method done by Sezer *et al.* [28] in Table 3.4. They used the same TILDA database and the quantitive performance evaluation in their work so the comparison would make sense.

In the quantitive performance evaluation, false alarm rate is given in the format as (***Number of sub-windows with false alarm : Total sub-windows***), shortly (***FW : TW***), and it is represented with the image folder ending with 'E0' that is full of defect free images. For a given textile type, we calculated 19200 total sub-windows of defect free images but Sezer *et al.* used 17250 sub-windows in their implementatiton so we normalize the false alarm rate obtained in their work. Correct classification evaluation is shown in the format as (***Missed images : Total images***), shortly (***MI : TI***), and it is represented with the defective image folders.

Table 3.4. Quantitive performance evaluation of the work done by Sezer *et al.*

	C1R1E0 **FW : TW**	**C1R1E1** **MI : TI**	**C1R1E2** **MI : TI**	**C1R1E3** **MI : TI**	**C1R1E4** **MI : TI**	**Total Miss** **MI : TI**
Sezer *et al.*	75 : 19200	0 : 50	0 : 50	2 : 50	1 : 50	3 : 200

3.5.1. Case 1: State and Observation Models are Linear, Their Noise Terms are Gaussian

If the system can be described in terms of this model representation, then in theory we know that Kalman filter gives the optimum solution. We will start with the Kalman filter and compare its results with other filter types.

3.5.1.1. Kalman Filter Approach. If we assume that our environment is linear and Gaussian, then we can write our state space model as

$$\mathbf{x}(t+1) = \mathbf{F}\mathbf{x}(t) + \mathbf{v}(t), \qquad \mathbf{v} \sim \mathcal{N}(\mathbf{0}, \mathbf{Q}) \tag{3.19}$$

$$\mathbf{y}(t) = \mathbf{H}\mathbf{x}(t) + \mathbf{n}(t), \qquad \mathbf{n} \sim \mathcal{N}(\mathbf{0}, \mathbf{R}) \tag{3.20}$$

Putting the parameter values in the correct places, we get **F** and **H** matrices as follows:

$$\mathbf{F} = \begin{bmatrix} a_1 & a_2 \\ 1 & 0 \end{bmatrix}$$

$$\mathbf{H} = \begin{bmatrix} 1 & 0 \end{bmatrix}$$

Here a_1 and a_2 are the 1-D AR coefficients that are estimated offline using the LS method that is discussed in section 3.2.1. Now, **F** and **H** matrices are available and using these matrices, we find the states $\mathbf{x}_k$ with the equations defined in section 2.3. The results obtained from the Kalman filter are given in Table 3.5 and some of the labeled images acquired from the implementation are shown in Appendix B.1.

3.5.1.2. EKF and UKF Approach. Actually EKF and UKF approaches are useful when the state space models are nonlinear. Here our models are linear so we implement these filters to show that the results obtained would not change much compared to Kalman filter.

1-D AR coefficients are again obtained offline with the same procedure as used in Kalman filter approach. For EKF implementation, we find the $\hat{\mathbf{F}}$ and $\hat{\mathbf{H}}$ matrices from the first order Taylor series expansion defined in Eqs. (2.17) and (2.18) which gives us

$$\hat{\mathbf{F}} = \begin{bmatrix} a_1 & a_2 \end{bmatrix}$$

$$\hat{\mathbf{H}} = \begin{bmatrix} 1 \end{bmatrix}$$

Using these matrices, we estimate the states $\mathbf{x}_k$ with the equations defined in section 2.3.1.

The implementation of UKF is started with the calculation of sigma points and their associated weights and ended with the update of the mean and covariance values for each of the observed data. We followed the procedure given in Appendix A.1.

3.5.1.3. Particle Filter Approach. In particle filter approach, we first initialize the state vector and find the predicted states using the state model given in Eq. (3.1). Here, we again use the AR parameters estimated using the LS method. As the new observation data becomes available, we calculate the weights using Eq. (2.38) because the importance density is taken as the transition prior. Since the observation noise is Gaussian, the weights are calculated using the Gaussian density function

$$w_k^i = \frac{1}{\sqrt{2\pi R}} \exp(-\frac{(y_k - x_k^i)^2}{2R})$$

where y_k is the new observation obtained at time k, x_k^i is the hidden state and R is the variance. Then resampling is done to reduce the effects of the degeneracy problem. The procedure we followed in resampling can be found in Appendix A.3.

In particle filter approach, the number of particles and the sub-window size are the two main factors that can affect the filter's performance. We tried different number of particles, N, ranging from 15 to 100 and different sub-window sizes ranging from 8x8 to 32x32 but we observed that the performance rates were not changing dramatically. So refering to the work done by Basibuyuk *et al.* [7], we found it suitable to take the number of samples N = 15 and the sub-window size as 32x32 after this time. The results obtained from the particle filter are given in Table 3.5 and some of the labeled images acquired from the implementation are shown in Appendix B.2.

3.5.1.4. Auxiliary Particle Filter Approach. In this approach, we have one more step compared to particle filter. The resampling stage is done at the previous time step so that the particles selected can be closer to the true state. The remaining steps are identical with particle filter approach. The procedure we have followed can be found in Appendix A.5.

The performance evaluation of all filter types discussed in Case 1 is given in Table 3.5.

Table 3.5. Quantitive performance evaluations for various filter types in Case 1

Filters	**C1R1E0** **FW : TW**	**C1R1E1** **MI : TI**	**C1R1E2** **MI : TI**	**C1R1E3** **MI : TI**	**C1R1E4** **MI : TI**	**Total Miss** **MI : TI**
Kalman	62 : 19200	0 : 50	2 : 50	1 : 50	6 : 50	9 : 200
EKF	61 : 19200	0 : 50	2 : 50	2 : 50	6 : 50	10 :200
UKF	62 : 19200	0 : 50	2 : 50	2 : 50	6 : 50	10 : 200
Particle	63 : 19200	0 : 50	2 : 50	1 : 50	6 : 50	9 : 200
Auxiliary	46 : 19200	12 : 50	2 : 50	7 : 50	7 : 50	28 : 200

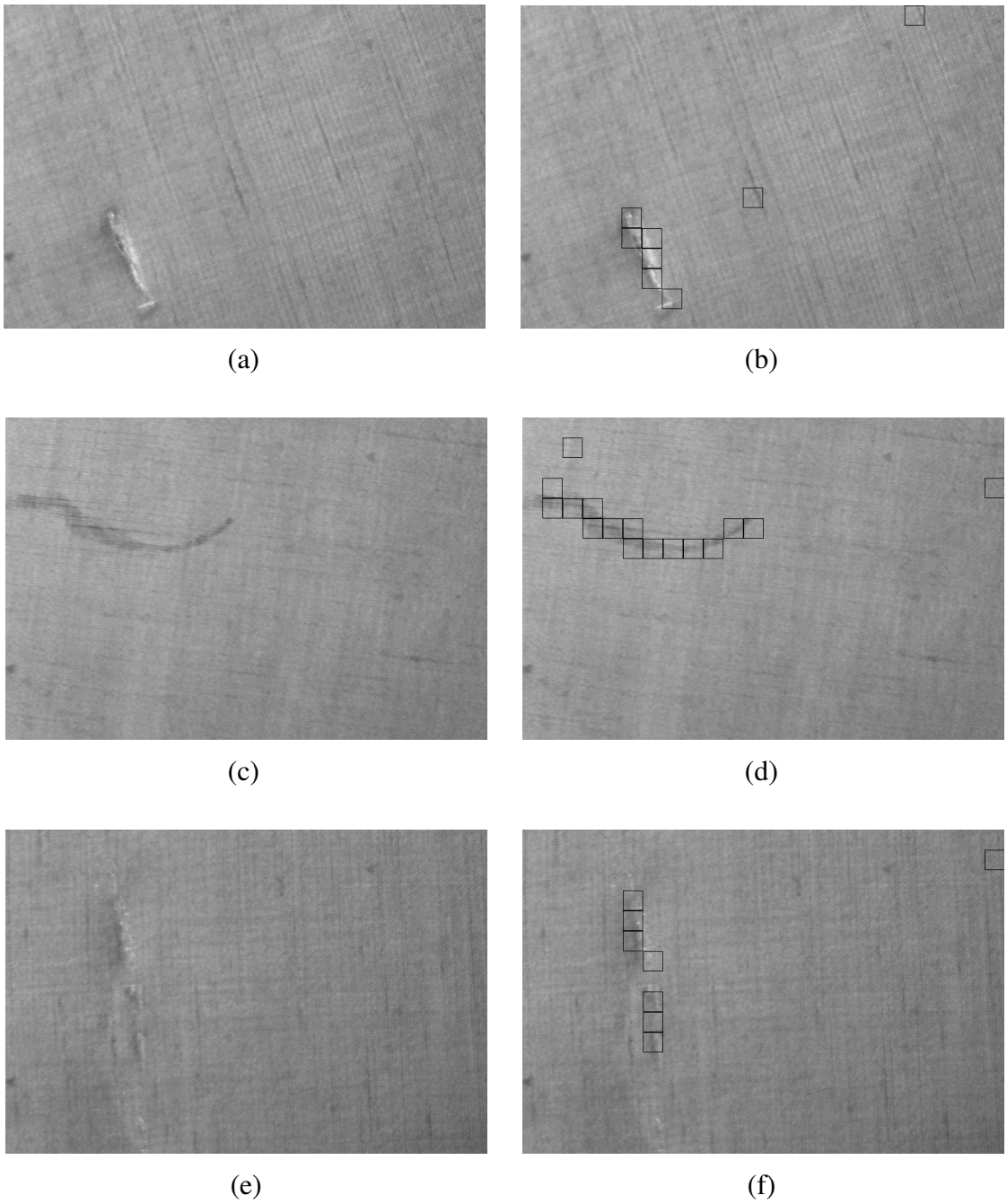
We display some of the sample pictures randomly selected from the filter implementation results in Fig. 3.4. The images on the left side of the figure displays the textile fabrics before the implementation and the ones on the right side shows the labeled situations after the implementation.

3.5.2. Case 2: State and Observation Models are Linear, State Noise is Gaussian but Observation Noise Term is non-Gaussian

From now on, particle filters will be used because Kalman filters are implemented for a linear and Gaussian environment case and they would not give better results for the cases that are going to be discussed.

As one class of non-Gaussian noise, αS family is used. The following equation is employed to calculate the weights in the update stage:

$$p(y_k \mid x_k^i) = \frac{1}{2\Pi} \int_{-\infty}^{+\infty} \exp\left\{ j\delta_k^i \zeta - \gamma_k^i \mid \zeta \mid^{\alpha_k^i} \left[1 + j\beta_k^i sign(\zeta)\omega(\zeta, \alpha_k^i)\right]\right\} \exp(j(y_k - x_k^i)\zeta) d\zeta$$



(a) (b)

(c) (d)

(e) (f)

Figure 3.4. An illustration of defect labeling on test images. Figures (a), (c), (e) are defective images; (b), (d) and (f) are their labeled situations, respectively.

The parameters that are utilized in the weight update equation are estimated using the resources [29] and [30] and they are found as

$$\begin{bmatrix} \alpha \\ \beta \\ \gamma \\ \delta \end{bmatrix} = \begin{bmatrix} 1.9 \\ 0 \\ 7 \\ 0 \end{bmatrix}$$

This reveals that the distribution is very close to a Gaussian distribution. In all our trials we observed that α was close to 2 and the distributions were not much heavy tailed as shown in Fig. 2.9. Thus, the implementation of the particle filter with this observation noise is done and the results did not change much.

3.5.3. Case 3: State and Observation Models are Linear, State Noise is non-Gaussian but Observation Noise Term is Gaussian

Since the state noise is non-Gaussian, 1-D AR coefficients are estimated using the IRLS algorithm. As a non-Gaussian state noise, αS and MoG noise terms are tried separately. Now the state model equation takes the form

$$\mathbf{x}(t) = 1.23\mathbf{x}(t-1) - 0.48\mathbf{x}(t-2) + \mathbf{v}(t)$$

When we regard the noise term as αS, we take the samples from the distribution that is generated with the parameters (α, β, γ, δ).

When we regard the noise term as MoG, we take the samples from the distribution that is formed with combining the Gaussian clusters as shown in Fig 2.10. In our case, the number of clusters is taken as n = 3 and MoG noise term is constructed from the samples that are drawn from the Gaussian distributions $\mathcal{N}(z_1, -9.5, 8.9)$, $\mathcal{N}(z_2, 7.2, \mathbf{7.6})$ and $\mathcal{N}(z_3, 2.3, \mathbf{4}.5)$, respectively.

The performance evaluation acquired running those two implementations are given in Table 3.6.

Table 3.6. Quantitive performance evaluations for particle filter whose observation noise is Gaussian

State noise	**C1R1E0** **FW : TW**	**C1R1E1** **MI : TI**	**C1R1E2** **MI : TI**	**C1R1E3** **MI : TI**	**C1R1E4** **MI : TI**	**Total Miss** **MI : TI**
MoG	58 : 19200	1 : 50	2 : 50	1 : 50	6 : 50	10 : 200
αS	58 : 19200	0 : 50	2 : 50	1 : 50	6 : 50	9 :200

3.5.4. Case 4: State and Observation Models are Linear, Their Noise Terms are non-Gaussian

Since the state noise is non-Gaussian in this case, we again estimate the 1-D AR parameters with IRLS algorithm and use the following state space model:

$$\mathbf{x}(t) = 1.23\mathbf{x}(t-1) - 0.48\mathbf{x}(t-2) + \mathbf{v}(t)$$

$$\mathbf{y}(t) = \mathbf{x}(t) + \mathbf{n}(t)$$

In case 2, we implemented the situation in which the observation noise **n**(t) was non-Gaussian and in case 3 the implementation is done when the state noise **v**(t) was non-Gaussian. Combining these two dynamics, we make both of the noise terms non-Gaussian.

For the non-Gaussian observation noise, the implementation results are shown in Table 3.7.

Table 3.7. Quantitive performance evaluations for particle filter whose observation noise is non-Gaussian

State noise	**C1R1E0** **FW : TW**	**C1R1E1** **MI : TI**	**C1R1E2** **MI : TI**	**C1R1E3** **MI : TI**	**C1R1E4** **MI : TI**	**Total Miss** **MI : TI**
MoG	60 : 19200	1 : 50	3 : 50	1 : 50	6 : 50	11 : 200
αS	60 : 19200	3 : 50	4 : 50	2 : 50	6 : 50	15 :200

3.5.5. Cases 5-8: State Model is 2-D Linear, Observation Model is Linear

In all of these cases from 5 to 8, we are using 2-D linear model defined in Eq. (2.47) as the state model. In this model type, the number of AR coefficients to be estimated is determined by the prediction mask size. We selected a 3x3 prediction mask and found eight 2-D AR parameters as shown in section 3.3.

The state and observation noise terms are defined in each case like 1-D linear model. In Table 3.8, all the results obtained from the filter implementations are given for 2-D linear state model and we display some of the labeled images acquired from the implementation are shown in Appendix B.3.

Table 3.8. Quantitive performance evaluations for particle filter in which the State Model is 2D linear, Observation Model is linear but Noise Terms change from Gaussian to non-Gaussian.

Cases	**C1R1E0** **FW : TW**	**C1R1E1** **MI : TI**	**C1R1E2** **MI : TI**	**C1R1E3** **MI : TI**	**C1R1E4** **MI : TI**	**Total Miss** **MI : TI**
Case 5	87 : 19200	0 : 50	1 : 50	1 : 50	6 : 50	8 :200
Case 6	88 : 19200	0 : 50	2 : 50	1 : 50	6 : 50	9 :200
Case 7	73 : 19200	0 : 50	2 : 50	1 : 50	6 : 50	9 :200
Case 8	75 : 19200	0 : 50	2 : 50	1 : 50	6 : 50	9 :200

So far from the performance evaluations, we can conclude that the textile fabric images are suitable for a linear and Gaussian state space model. The data does not contain much outlier so Kalman filter types can give the optimum. Now we will try to add noise to our images to make the tails heavier as displayed in Figure 2.9 and then we compare the performances of the filters as a response to this change.

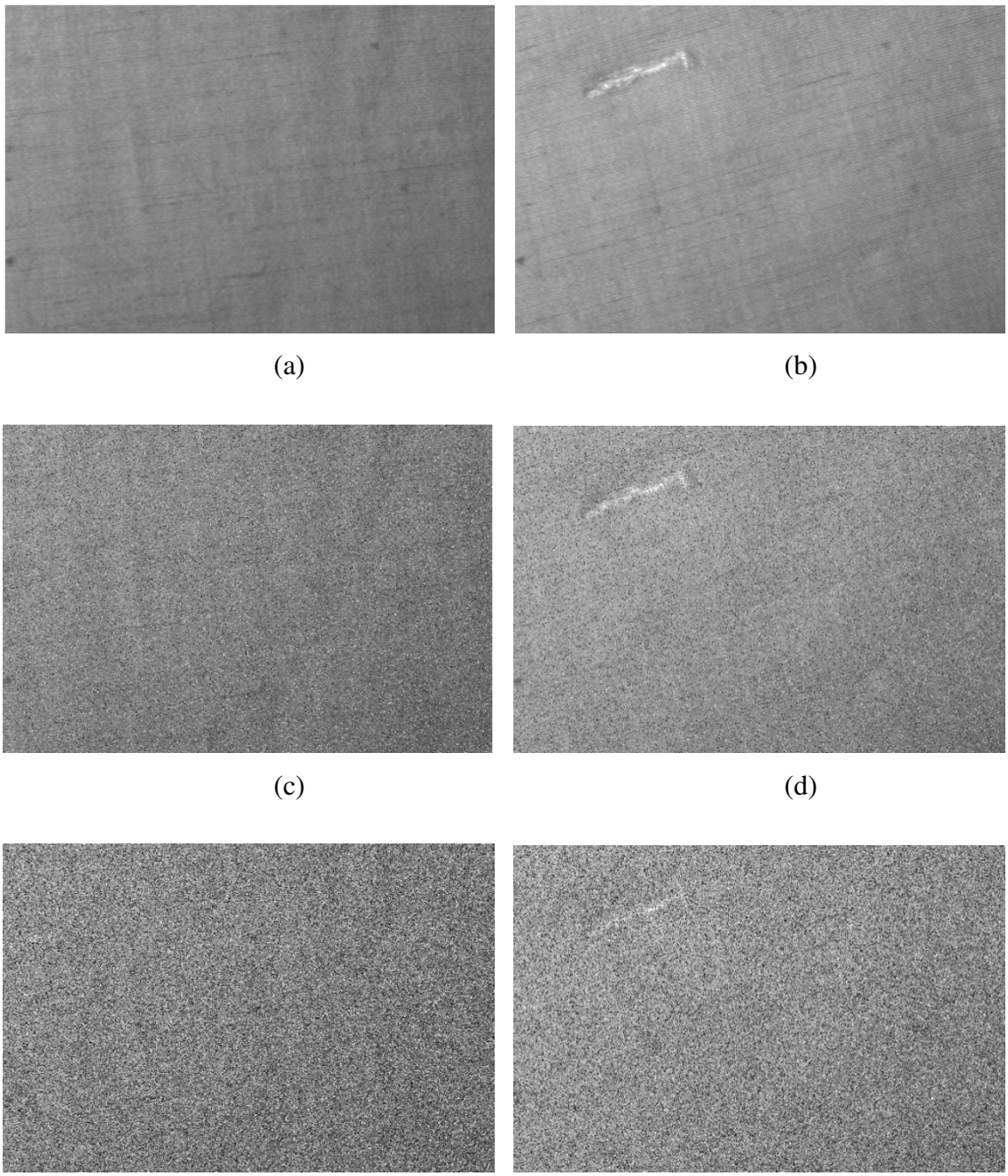
3.6. Noise Addition to the Images

To represent the outlier in the textile fabric image, we supposed that salt and pepper noise would be a good candidate for this purpose. Salt and pepper noise generally occurs when the camera sensors have a problem and when it causes them to malfunction. If the image is represented with 8 bits, then the intensity value for salt and pepper noises are 255 and 0, respectively.

In Fig. 3.5, we show the original images and the ones that include salt and pepper noise with specific noise densities. The noise density means that the specified proportion of the pixels are noisy in the image. The distribution between salt and pepper is fifty per cent.

After the addition of salt and pepper noise to the fabric textile images, it is assumed that the images cannot be modeled with Gaussian distribution anymore. Now the images contain impulsive noise and it should be represented with a non-Gaussian noise. According to this fact, we decided to try our filter types on this data and see the changes in the results.

While running the algorithms, we took the noise density as 0.1. First we implemented the Kalman filter for Case 1 and ignored other Kalman filter types because any change in the system nonlinearity did not occur. Then, we run the particle filter for Case 3 and Case 4 with different number of particles. In section 3.5, we concluded that the performance evaluations were almost same for N = 15 and N = 100 without adding the noise to the images. However, we observed that an improvement in the performance can occur if the number of particles is increased in the noisy environment because it is expected to approximate the posterior density more accurately when the system is non-Gaussian. The performance evaluations we obtained are given in Table 3.9 and some of the labeled images acquired from the implementation are shown in Appendix B.4.



(a) (b)

(c) (d)

(e) (f)

Figure 3.5. Defect free image and the defective image are shown in (a) and (b). Their noisy situations with noise densities 0.1 and 0.3 are shown in (c), (d) and (e), (f), respectively.

Table 3.9. Quantitive performance evaluations when noise is added to textile fabric images

Filters	**C1R1E0** **FW : TW**	**C1R1E1** **MI : TI**	**C1R1E2** **MI : TI**	**C1R1E3** **MI : TI**	**C1R1E4** **MI : TI**	**Total Miss** **MI : TI**
Kalman	3 : 19200	38 : 50	50 : 50	43 : 50	36 : 50	167 : 200
Particle (Case 3)	66 : 19200	1 : 50	4 : 50	4 : 50	9 : 50	18 : 200
Particle (Case4)	25 : 19200	14 : 50	12 : 50	18 : 50	17 : 50	61 : 200

3.7. Time-performance Analysis

The implementation of the algorithms is run on a notebook that has Intel Core2 2.00GHz CPU and 1.00 GB RAM. The speed of the implementation was affected with the sub-window size and the number of samples for particle filter case. We decided to use the window size as 32x32 and the number of samples as 15 to achieve the optimum performance rate which was constrained by the time of the operation and the performance of the filter. In Fig. 3.9, we show the total number of missed images, number of sub-windows with false alarms and the time of the operation spent for each image.

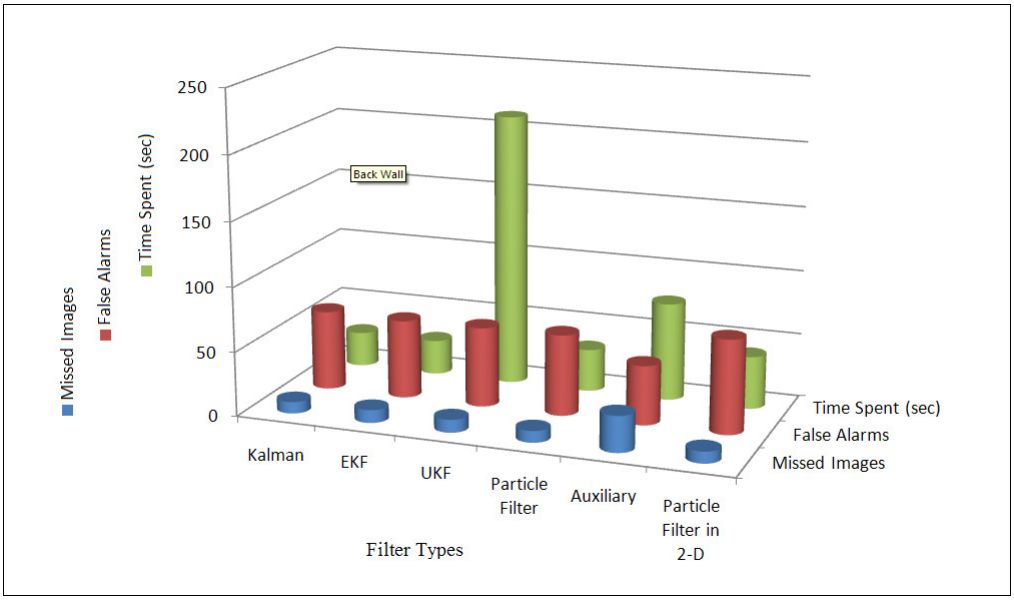


Figure 3.6. Time-performance analysis of various filter types

4. CONCLUSIONS AND DISCUSSIONS

Various filter type implementations are explained and their performance evaluations are given in implementation section. In all of this work, we tried to determine the characteristics of the textile fabric images and find the best filter for these images in terms of both achieved performance rate and the operation time spent to run the algorithms.

From Table 3.2, we concluded that IRLS algorithm gives more accurate results than the LS algorithm if the data contains non-Gaussian noise. Thus, we expected to have higher achievement rates when the filters run in a non-Gaussian environment. However, we obtained almost same results when the state noise term was changed from Gaussian to non-Gaussian as can be seen in Tables 3.5 and 3.6. Before running the filters, we supposed that αS noise would be a good candidate to model textile fabric images but its estimated parameters showed us that the distribution adapting to these images was very similar to a Gaussian distribution. It induced us to assume that the state model noise term is close to a Gaussian distribution. On the other hand, making the observation noise term non-Gaussian decreased the performance rates as shown in Tables 3.7 and 3.9. It also makes us to think that the observation noise term to be Gaussian is highly probable.

From the results in Table 3.5, we observed that the Auxiliary particle filter does not give better results than the particle filter. In theory, it can give better results if the likelihood is located in the tails of the prior so from our performance evaluations we can conclude that the likelihood and the prior coincides in our system.

In Tables 3.5 and 3.8, we can see that Kalman filter and their versions of EKF and UKF give almost same success rates with the particle filters that are run in both 1-D and 2-D linear state model. It shows us that the textile fabric images are suitable to represent them in a linear and Gaussian system environment.

However, from the theoretical definitions and practical implementations we know that particle filters outperform Kalman filters if the system is nonlinear and non-Gaussian. To make the system similar to a non-Gaussian one, we decided to include outlier data in the images and consequently we added salt and pepper noise to them. Since it makes the tails heavier in a Gaussian distribution, it is no more convenient to represent the images with a Gaussian noise density. The results obtained in this manner in Table 3.9 showed us that the particle filter can yield better results than the Kalman filter. What is more, it should be noted that the performances of all filter types have dropped due to noise effect.

Also we can conclude that without adding noise to the images, Kalman filter would be the optimal filter type to use in this environment because it can give highly accurate results as other filter types but less operation time is needed to run it.

Finally, we can conclude that the Bayesian approach to textile defect detection problem does not give high defect detection rates as the texture analysis method of Sezer *et al.* [28] but it gives less false alarm rates compared to it.

4.1. Future Work

To model the fabric textile images in a state space model, we looked for finding a suitable texture model and a noise type that fits well to these images. For this purpose, we tried several approaches that exist in the literature. It would be a good improvement if a nonlinear texture model can be developed to describe the fabric textile images. However it should be noticed that the technique should be causal since filtering deals with the past data.

Also in particle filter implementation, we were using the transition prior as proposal density. There can be done more study on the filter to make the proposal density better. As the proposal density coincides with the likelihood more and more, the performance of the filter increases.

Finally one of the most important things that should be improved is the speed of the algorithms that are implemented. We tried to make them as fast as possible without dropping the success rate but there is still need to work on it.

APPENDIX A: ALGORITHMS

A.1. Unscented Kalman Filter Algorithm

Initialize the state vector $\mathbf{x}$ and covariance matrix $\mathbf{P}_x$.

for $k = 1$ *to* T

Calculate sigma points $\mathbf{X}_k$ and their weights W_i with (2.25) and (2.26).

Use the state model to find

$$\mathbf{X}_{k+1} = \mathbf{f}(\mathbf{X}_k, \mathbf{v}_k)$$

Find the predicted mean and covariance

$$\bar{\mathbf{x}}_{k+1} = \sum_{i=0}^{2n} W_i \mathbf{X}_{i,k+1}$$

$$\mathbf{P}_{xx,k+1} = \sum_{i=0}^{2n} W_i (\mathbf{X}_{i,k+1} - \bar{\mathbf{x}}_{k+1})(\mathbf{X}_{i,k+1} - \bar{\mathbf{x}}_{k+1})^T$$

Use the observation model to find

$$\mathbf{Y}_{k+1} = \mathbf{h}(\mathbf{X}_{k+1}, \mathbf{n}_k)$$

Find the predicted mean and covariance

$$\bar{\mathbf{y}}_{k+1} = \sum_{i=0}^{2n} W_i \mathbf{Y}_{i,k+1}$$

$$\mathbf{P}_{yy,k+1} = \mathbf{R}_{k+1} + \sum_{i=0}^{2n} W_i (\mathbf{Y}_{i,k+1} - \bar{\mathbf{y}}_{k+1})(\mathbf{Y}_{i,k+1} - \bar{\mathbf{y}}_{k+1})^T$$

Find the cross correlation matrix

$$\mathbf{P}_{xy,k+1} = \sum_{i=0}^{2n} W_i (\mathbf{X}_{i,k+1} - \bar{\mathbf{x}}_{k+1})(\mathbf{Y}_{i,k+1} - \bar{\mathbf{y}}_{k+1})^T$$

Find the new state vector and covariance matrix

$$\mathbf{K}_{k+1} = \mathbf{P}_{xy,k+1} \mathbf{P}_{yy,k+1}^{-1}$$

$$\mathbf{x}_{k+1} = \mathbf{x}_k + \mathbf{K}_{k+1}(\mathbf{y} - \bar{\mathbf{y}})$$

$$\mathbf{P}_x = \mathbf{P}_{xx,k+1} - \mathbf{K}_{k+1} \mathbf{P}_{yy,k+1} \mathbf{K}_{k+1}^T$$

Return to calculate new sigma points

end

A.2. SIS Particle Filter Algorithm

$$\{\mathbf{x}_k^i, w_k^i\}_{i=1}^N = SIS\left[\{\mathbf{x}_{k-1}^i, w_{k-1}^i\}_{i=1}^N, \mathbf{y}_k\right]$$

for i = 1 : N

- Draw $\mathbf{x}_k^i \sim q(\mathbf{x}_k \mid \mathbf{x}_{k-1}^i, \mathbf{y}_k)$
- Find the importance weights $\tilde{w}_k^i \propto w_{k-1}^i \dfrac{p(\mathbf{y}_k \mid \mathbf{x}_k^i)\, p(\mathbf{x}_k^i \mid \mathbf{x}_{k-1}^i)}{q(\mathbf{x}_k^i \mid \mathbf{x}_{k-1}^i, \mathbf{y}_k)}$

end

- Calculate the total weight $W = sum(\tilde{w}_k^i)_{i=1}^N$

for i = 1 : N

- Normalize the weights $w_k^i = \dfrac{\tilde{w}_k^i}{W}$

end

A.3. Resampling Algorithm

$$\{\mathbf{x}_k^{j*}, w_k^j, i^j\}_{j=1}^N = resample\left[\{\mathbf{x}_k^i, w_k^i\}_{i=1}^N\right]$$

- Initialize the cumulative sum of weights (CSW) $W_1 = w_k^1$

for $i = 2 : N$

- Construct the CSW $W_i = W_{i-1} + w_k^i$

end

- Initialize $i = 1$ and draw a random starting point $u_1 = \frac{rand}{N}$

for $j = 1 : N$

- Move along the CSW $u_j = u_1 + \frac{j-1}{N}$

while $u_j > W_i$

  - $i = i + 1$

end

- Assign sample, weight and index $\mathbf{x}_k^{j*} = \mathbf{x}_k^i$, $w_k^j = \frac{1}{N}$, $i^j = i$

end

A.4. SIR Particle Filter Algorithm

$$\{\mathbf{x}_k^i, w_k^i\}_{i=1}^N = SIR\left[\{\mathbf{x}_{k-1}^i, w_{k-1}^i\}_{i=1}^N, \mathbf{y}_k\right]$$

for i = 1 : N

- Draw $\mathbf{x}_k^i \sim p(\mathbf{x}_k \mid \mathbf{x}_{k-1}^i)$
- Find the importance weights $\tilde{w}_k^i \sim p(\mathbf{y}_k \mid \mathbf{x}_k^i)$

end

- Calculate the total weight $W = sum(\tilde{w}_k^i)_{i=1}^N$

for i = 1 : N

- Normalize the weights $w_k^i = \frac{\tilde{w}_k^i}{W}$

end

- Resample the particles $\{\mathbf{x}_k^i, w_k^i, -\}_{i=1}^N = resample\left[\{\mathbf{x}_k^i, w_k^i\}_{i=1}^N\right]$

A.5. ASIR Particle Filter Algorithm

$$\{\mathbf{x}_k^i, w_k^i\}_{i=1}^N = ASIR\left[\{\mathbf{x}_{k-1}^i, w_{k-1}^i\}_{i=1}^N, \mathbf{y}_k\right]$$

for i = 1 : N

- Draw $\boldsymbol{\tau}_k^i \sim p(\mathbf{x}_k \mid \mathbf{x}_{k-1}^i)$
- Find the importance weights $\tilde{w}_k^i \propto w_{k-1}^i p(\mathbf{y}_k \mid \mathbf{x}_k^i)$

end

- Calculate the total weight $W = sum(\tilde{w}_k^i)_{i=1}^N$

for i = 1 : N

- Normalize the weights $w_k^i = \dfrac{\tilde{w}_k^i}{W}$

end

- Resample the particles $\{-,-,i^j\}_{i=1}^N = resample\left[\{\mathbf{x}_k^i, w_k^i\}_{i=1}^N\right]$

for i = 1 : N

- Draw $\mathbf{x}_k^i \sim p(\mathbf{x}_k \mid \mathbf{x}_{k-1}^{i^j})$
- Find the importance weights $\tilde{w}_k^i \propto \dfrac{p(\mathbf{y}_k \mid \mathbf{x}_k^i)}{p(\mathbf{y}_k \mid \boldsymbol{\mu}_k^{i^j})}$

end

- Calculate the total weight $W = sum(\tilde{w}_k^i)_{i=1}^N$

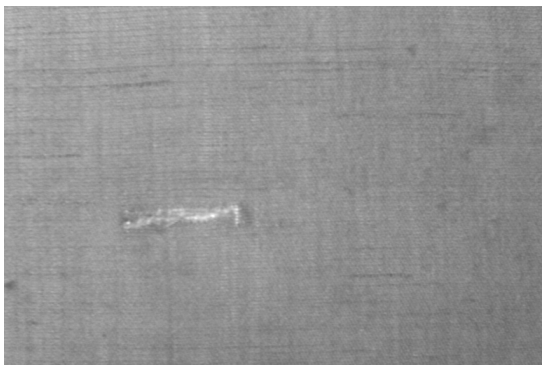
for i = 1 : N

- Normalize the weights $w_k^i = \dfrac{\tilde{w}_k^i}{W}$

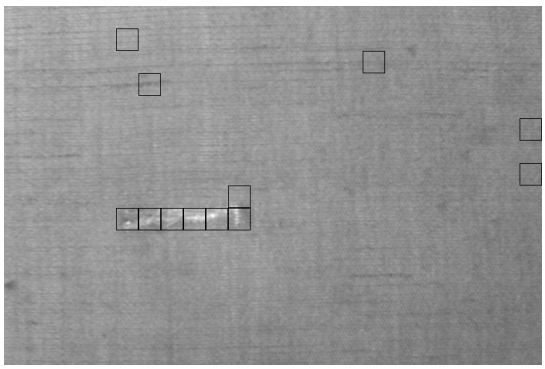
end

APPENDIX B: SAMPLES FROM THE IMPLEMENTATIONS

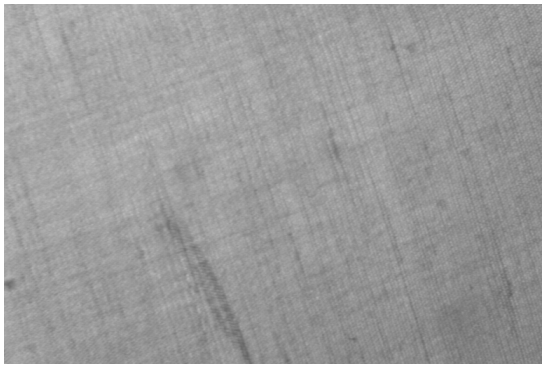
B.1. Samples from the Kalman Filter Implementation



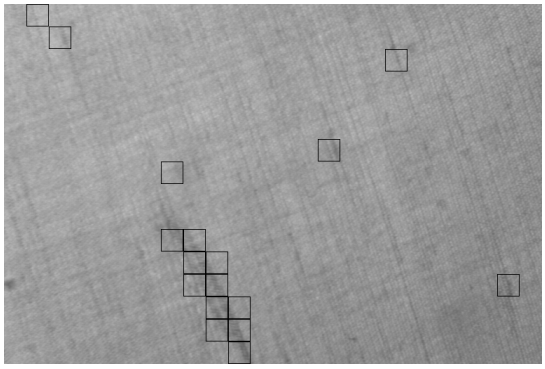
(a) “C1R1E1\C1R1EACB.TIF”



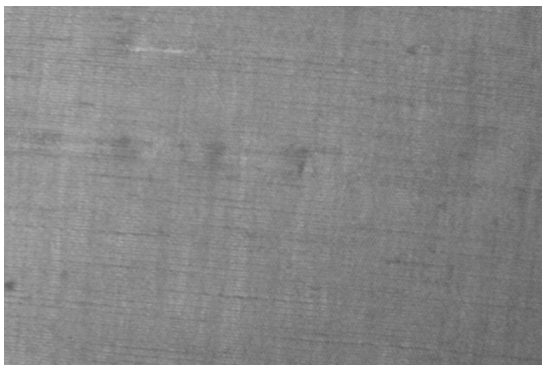
(b) Labeled defective image in (a).



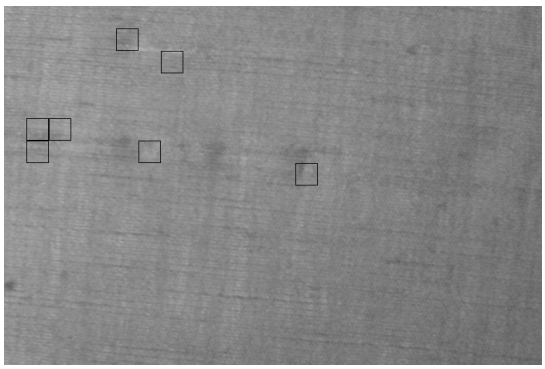
(c) “C1R1E2\C1R1E2N6.TIF”



(d) Labeled defective image in (c).

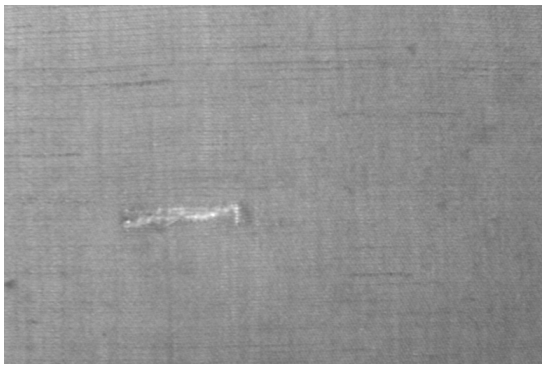


(e) “C1R1E3\C1R1EAHX.TIF”

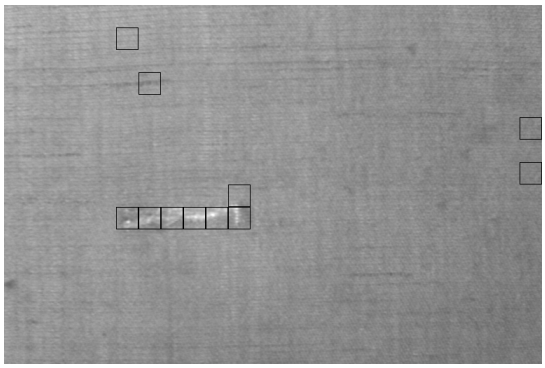


(f) Labeled defective image in (e).

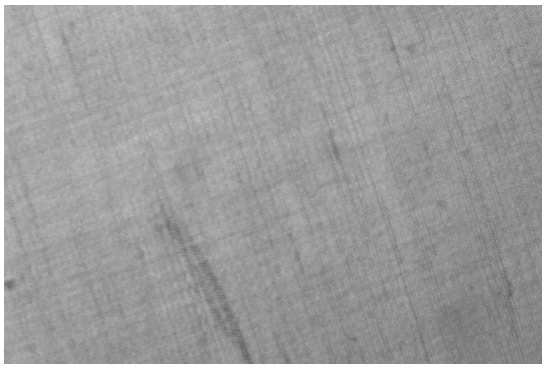
B.2. Samples from the Particle Filter Implementation



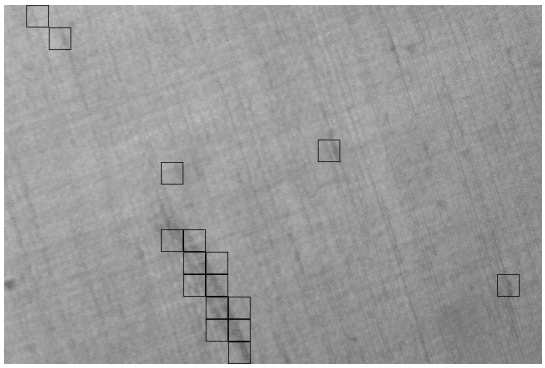
(a) “C1R1E1\C1R1EACB.TIF”



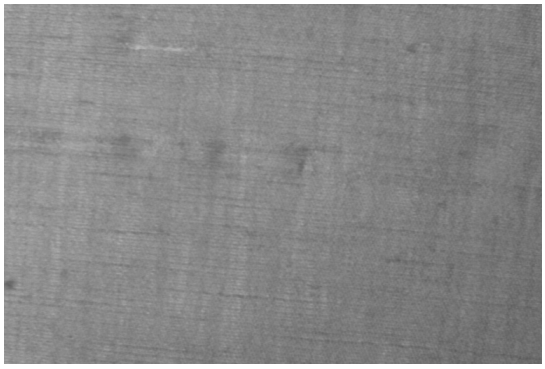
(b) Labeled defective image in (a).



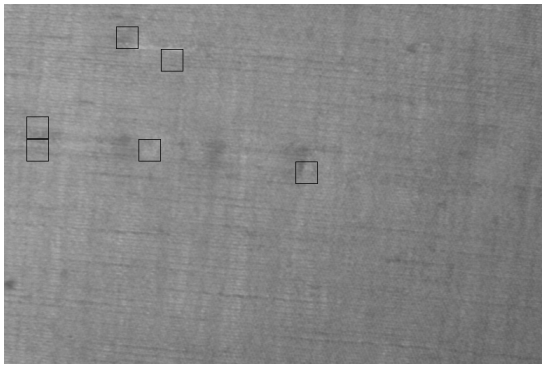
(c) “C1R1E2\C1R1E2N6.TIF”



(d) Labeled defective image in (c).

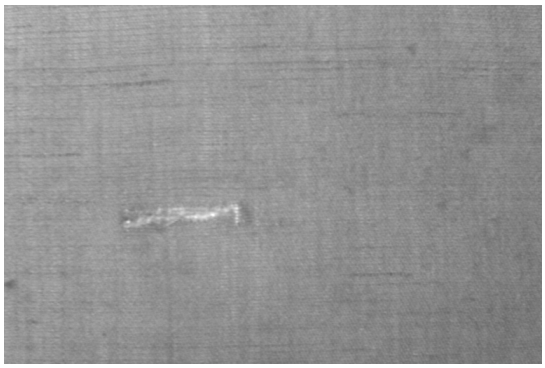


(e) “C1R1E3\C1R1EAHX.TIF”

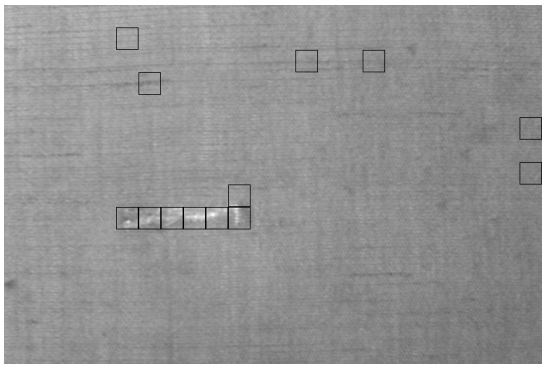


(f) Labeled defective image in (e).

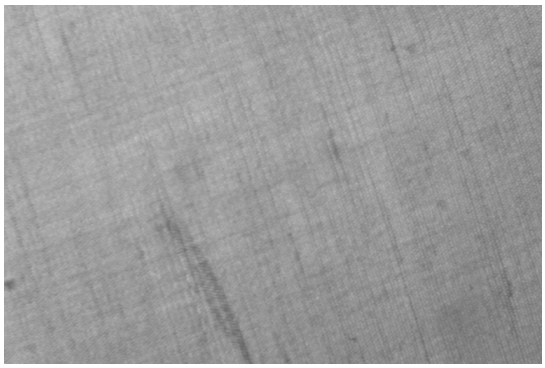
B.3. Samples from the Particle Filter Implementation in 2-D



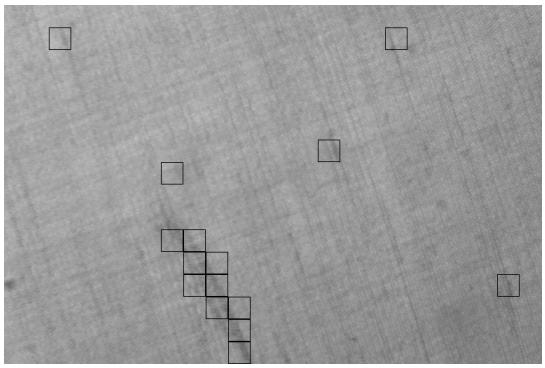
(a) “C1R1E1\C1R1EACB.TIF”



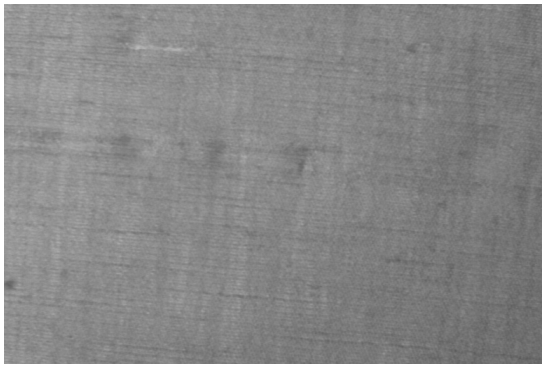
(b) Labeled defective image in (a).



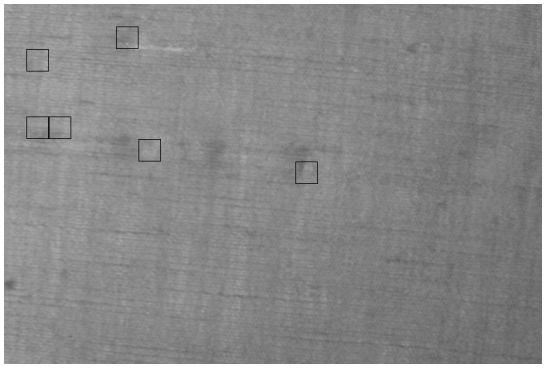
(c) “C1R1E2\C1R1E2N6.TIF”



(d) Labeled defective image in (c).

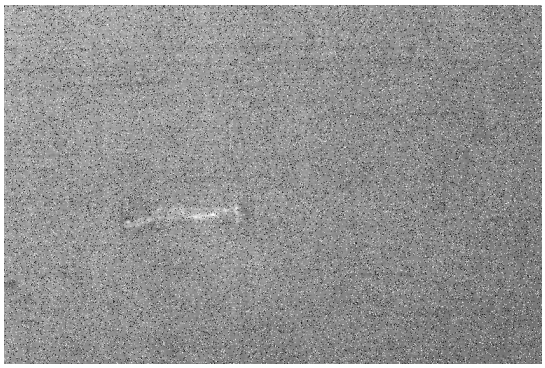


(e) “C1R1E3\C1R1EAHX.TIF”

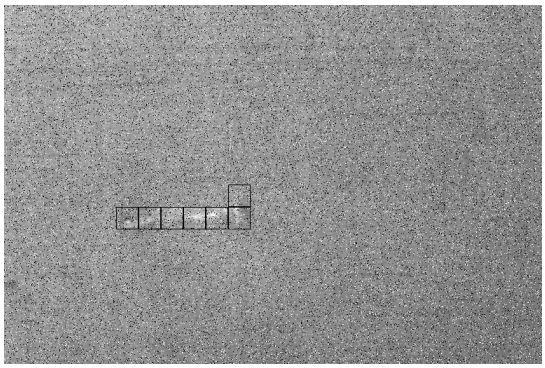


(f) Labeled defective image in (e).

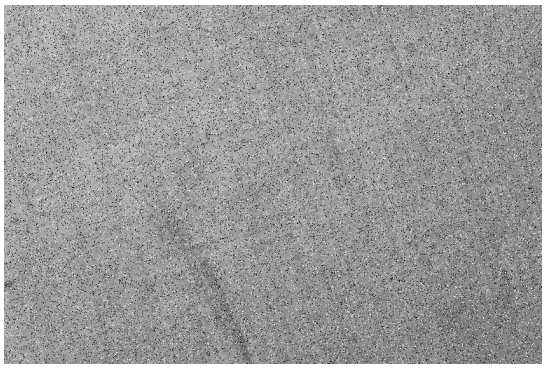
B.4. Samples from the Particle Filter Implementation with Noisy Images



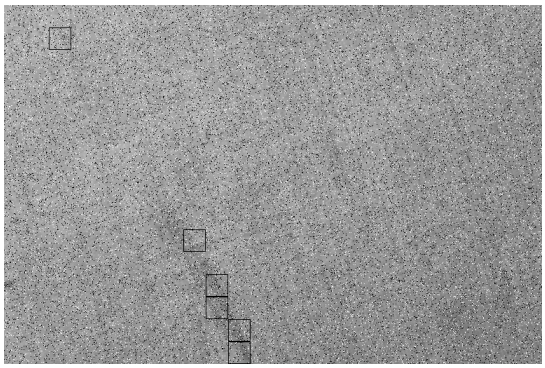
(a) Noisy "C1R1E1\C1R1EACB.TIF"



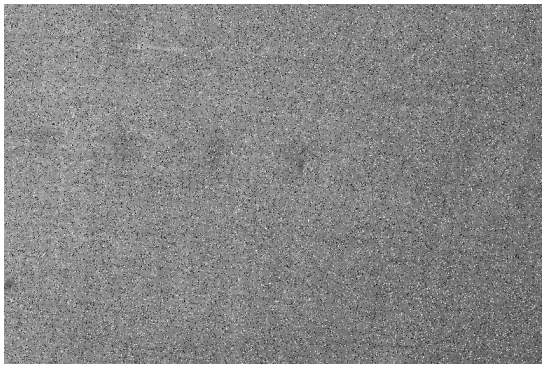
(b) Labeled defective image in (a).



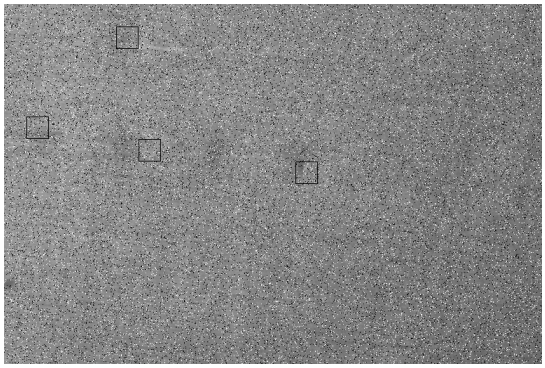
(c) Noisy "C1R1E2\C1R1E2N6.TIF"



(d) Labeled defective image in (c).



(e) Noisy "C1R1E3\C1R1EAHX.TIF"



(f) Labeled defective image in (e).

REFERENCES

1. Jasper, W., Joines, J. A. and Brenzovich, J., “Fabric Defect Detection using a GA Tuned Wavelet Filter”, *Computers and Their Applications*, pp. 345-350, 2003.

2. Chen, J. and Jain, A. K., "A structural approach to identify defects in textured images", *Proceedings of IEEE International Conference on Systems, Man, and Cybernetics (SMC '88)*, Vol. 1, pp. 29-32, August 1988.

3. Aras, B., Ertüzün, A. and Erçil, A., “Higher order statistics based texture analysis method for defect inspection of textile products”, *Proceedings of the IEEE-EURASIP Workshop on Nonlinear Signal and Image Processing (NSIP99)*, pp. 858-862, 20-23 June 1999.

4. Latif-Amet, A., Ertüzün, A. and Erçil, A., “An efficient method for texture defect detection: subband domain co-occurrence matrices”, *Image Vision Computing,* Vol. 18, pp. 543-553, May 2000.

5. Yang, X., Pang, G. and Yung, N., “Robust fabric defect detection and classification using multiple adaptive wavelets”, *IEE Proceedings - Vision, Image and Signal Processing,* 152(6), pp. 715—723, 2005.

6. Kumar, A. and Pang, G. K. H., "Defect detection in textured materials using optimized filters," *IEEE Transactions on Systems, Man, and Cybernetics, Part B*, Vol. 32, No. 5, pp. 553-570, 2002.

7. Başıbüyük, K., Çoban, K. and Ertüzün, A., “Model Based Defect Detection Problem: Particle Filter Approach”, *3rd IEEE International Symposium on Communications, Control and Signal Processing (ISCCSP 2008)*, March 12-14, 2008.

8. Arulampalam, M. S., Maskell, S., Gordon, N., and Clapp, T., “A tutorial on particle filters for online nonlinear/non-Gaussian Bayesian tracking”, *IEEE Trans. Sig. Proc. 50, 2 (February)*, pp. 174-188, 2002.

9. Gordon, N., Salmon, D., and Smith, A., "Novel approach to nonlinear/non-Gaussian Bayesian state estimation," *IEE Proceedings on Radar and Signal Processing*, Vol. 140, pp. 107-113, 1993.

10. Ristic, B., Arulampalam, S., Gordon, N., *Beyond the Kalman Filter*, Artech House, 2004.

11. van der Merwe, R., Doucet, A., de Freitas, N. and Wan, E., “The Unscented Particle Filter”, *Advances in Neural Inf. Processing Systems (NIPS13)*, pp. 584-590, 2001.

12. Julier, S. and Uhlmann, J., “A new extension of the Kalman filter to nonlinear systems”, *Int. Symp. Aerospace/Defense Sensing, Simul. and Controls*, Orlando, FL, USA, 1997.

13. Doucet, A., de Freitas, N., and Gordon, N., “Sequential Monte Carlo Methods in Practice”, *Statistics for engineering and information science*, Springer-Verlag, Berlin, 2001.

14. Oudjane, N. and Musso, C., "Progressive correction for regularized particle filters", *Proc. IEE Proceedings of the 3rd International Conference on Information Fusion*, Vol. 2, Paris, France, July 2000.

15. Gilks, W. R. and Berzuini, C., “Following a moving target--Monte Carlo inference for dynamic Bayesian models”, *J. R. Stat. Soc. Ser. B 63*, pp. 127-146, 2001.

16. Mühlich, M., “Particle Filters an Overview”, *Talk at Filter Workshop*, Bucharest,Romania, March 2003.

17. Geman, S. and Geman, D., "Stochastic relaxation, Gibbs distributions, and the Bayesian restoration of images", *Readings in uncertain reasoning*, pp. 452-472, Morgan Kaufmann Publishers Inc., San Francisco, CA, 1990.

18. Winkler, G., *Image Analysis, Random Fields and Markov Chain Monte Carlo Methods,* Springer, 2006.

19. Cross, G. R. and Jain, A. K., "Markov random field texture models", *IEEE Trans. Pattern. Anal. and Machine Intell.*, Vol. PAMI-5, No. 1, pp. 25-39, 1983.

20. Wikipedia, [Online], http://en.wikipedia.org/wiki/Normal_distribution, [Cited: 5 30, 2010].

21. Kuruoglu, E. and Zerubia, J., "Skewed alpha-stable distributions for modeling textures", *Pattern Recognition Letters*, Vol. 24, No. 1-3, pp. 339-348, January 2003.

22. Penny, W., "Variational Bayes for d-dimensional Gaussian mixture models", *Technical report*, Wellcome Department of Cognitive Neurology, University College London, 2001.

23. Workgroup on Texture Analysis of DFG.TILDA Textile Texture Database, [Online], (http://lmb.informatik.uni-freiburg.de/research/dfg-texture/tilda), [Cited: 5 30, 2010].

24. Mohit, *G.,* "Linear Prediction Algorithms*",* Indian Institute of technology, Bombay, India, April 2003.

25. Maronna, R. A., Martin, D. and Yohai, V. J., *Robust Statistics: Theory and Methods,* Wiley, 2006.

26. Erçil, A., "Robust Ridge Regression", *Research Publication*, General Motors Research Labs, GMR-5349, 1986.

27. Maragos, P. A., Schafer, R. W. and Mersereau, R. M., "Two-dimensional linear prediction and its application to adaptive predictive coding of images", *IEEE Trans. Acoust., Speech, Signal Processing*, vol. ASSP-32, pp. 1213-1229, 1984.

28. Sezer, O. G., Ercil, A. and Ertuzun, A., "Using perceptual relation of regularity and anisotropy in the texture with independent component model for defect detection", *Pattern Recognition*, Vol. 40, No. 1, pp.121-133, January, 2007.

29. Kuruoglu, E., "Density Parameter Estimation of Skewed Alpha-Stable Distributions", *IEEE Transactions on Signal Processing*, Vol. 49, No. 10, pp. 2192-2201, 2001.

30. Mark Veillette, [Online], http://math.bu.edu/people/mveillet/html/alphastablepub.html, [Cited: 5 30, 2010].