DOKUZ EYLÜL UNIVERSITY
GRADUATE SCHOOL OF NATURAL AND APPLIED SCIENCES

IMAGE REGISTRATION USING ARTIFICIAL NEURAL NETWORKS

by

Devin SAĞIRLIBAŞ

October, 2010

İZMİR

IMAGE REGISTRATION USING ARTIFICIAL NEURAL NETWORKS

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

by

Devin SAĞIRLIBAŞ

October, 2010

İZMİR

M.Sc THESIS EXAMINATION RESULT FORM

We have read the thesis entitled "**IMAGE REGISTRATION USING ARTIFICIAL NEURAL NETWORKS**" completed by **DEVİN SAĞIRLIBAŞ** under supervision of **ASSIST. PROF. DR. YAVUZ ŞENOL** and we certify that in our opinion it is fully adequate, in scope and in quality, as a thesis for the degree of Master of Science.

Assist. Prof. Dr. Yavuz ŞENOL

Supervisor

Assist. Prof. Dr. Haldun SARNEL	Assist. Prof. Dr. Gökhan DALKILIÇ
(Jury Member)	(Jury Member)

Prof. Dr. Mustafa SABUNCU

Director

Graduate School of Natural and Applied Sciences

ACKNOWLEDGMENTS

I would like to thank to my advisor Asst. Prof. Dr. Yavuz ŞENOL for his support and valuable guidance during the course of this thesis.

I'm very grateful to Asst. Prof. Dr. Haldun Sarnel for his help, sharing his knowledge and guiding me in this thesis.

I would like to thank to my parents for their encouragement and endless love.

My special thanks are for the most important and precious members of my family, my husband Mehmet Volkan SAĞIRLIBAŞ and my little son Sarp SAĞIRLIBAŞ.

I wish to express my special thanks to my husband Volkan for his endless support, love, care, patience as a husband and also specially thank to him for sharing his professional experience with me as a colleague.

I would also like to express my special thanks to my son Sarp for his patience during my long working hours and calming me when I was disappointed.

Devin SAĞIRLIBAŞ

IMAGE REGISTRATION USING ARTIFICIAL NEURAL NETWORKS

ABSTRACT

Image registration is a procedure that transforms different sets of data that are multiple photographs or data from different sensors, from different times, or from different viewpoints into one coordinate system. In this thesis affine transform is chosen as the transform model.

In this thesis image registration process is done using neural networks in the presence of noise. In the applications, we had three images and affine transform was applied all of the images. At first, features were extracted from the images and these features were given to the network as inputs, then estimated parameters were obtained at the output. These features were extracted by the methods of discrete cosine transform (DCT) and two dimensional principal component analysis (2DPCA). In a pre-registration phase, extracted features from a set of translated, rotated and scaled images of the same scene are employed to train both a Radial Basis Function Neural Network (RBF NN) and a Feed-forward Neural Network (FNN). In the registration phase, the features are extracted from the test image and these features are given to the network. By this way, registration parameters are obtained.

The results were compared both according to the different feature extraction methods and different type of neural networks.

Keywords: Image Registration, Radial Basis Function Neural Network, Feed-forward Neural Network, DCT Coefficients, 2DPCA

YAPAY SİNİR AĞLARI KULLANARAK GÖRÜNTÜ ÇAKIŞTIRMA

ÖZ

Görüntü çakıştırma çeşitli sensörlerden, değişik zamanlarda ya da farklı bakış açılarından çekilen birçok fotoğraf olarak tanımlanabilen çeşitli gruplardaki veriyi bir koordinat sistemine dönüştüren bir işlemdir. Bu tezde, dönüştürme modeli olarak afin dönüşümü seçilmiştir.

Bu tezde görüntü çakıştırma işlemi sinir ağları kullanılarak gürültünün varlığında yapılmaktadır. Uygulamalarda üç tane resmimiz bulunmaktadır ve bu resimlerin hepsine afin dönüşümü uygulanmıştır. İlk olarak resimlerden özellikler çıkarmış ve bu özellikler ağa giriş olarak verilmiş sonra çıkışta tahmini parametreler elde edilmiştir. Bu özellikler ayrık kosinüs dönüşünü ya da iki boyutlu temel bileşen analizi ile çıkarıldı. Çakıştırma öncesi evrede aynı manzaraya ait ötelenmiş, döndürülmüş ve ölçeklendirilmiş resimlerden çıkarılmış özellikler hem radyal tabanlı yapay sinir ağını hem de ileri beslemeli sinir ağını eğitmek için kullanılmıştır. Çakıştırma evresinde, test resminden özellikler çıkarılır ve bu özellikler ağa verilir. Bu şekilde çakıştırma parametreleri elde edilir.

Sonuçlar hem çeşitli özellik çıkarma yöntemlerine göre hem de yapay sinir ağı çeşitlerine göre karşılaştırılmıştır.

Anahtar Kelimeler: Görüntü Çakıştırma, Radyal Tabanlı Yapay Sinir Ağı, İleri Beslemeli Sinir Ağı, DCT Katsayıları, 2 Boyutlu Temel Bileşen Analizi

CONTENTS

Page

CHAPTER ONE

INTRODUCTION

1.1 Introduction

Image registration is a technique that determines the spatial best fit between two images which overlaps the same screen. That is; by image registration two images that are taken in different times with different sensor from different viewpoints can be matched. In many image processing applications, registration is a fundamental stage because of this property. The application areas of image registration are computer vision and pattern recognition, medical image analysis, remote sensing, image matching-based vehicle guidance and super-resolution.

According to the essential ideas of the registration algorithm criterion, most image registration methods are divided into feature-based and area based methods that are which will be discussed in the Chapter Two-Image Registration Theory in detail. In this thesis, neural network-based image registration was used as the registration method. A neural network is trained with a set of global image features at inputs that represents an image transformed by some known parameters and the known parameters at the outputs. After the training process, when its features of the same type are input to the network, the trained neural network can estimate unknown parameters of a query image.

Image registration is done with various methods and one of the newest methods used in image registration is neural network approach. In an early study (Qian, & Li, 1997), Hopfield neural network was used. This network was used only in matching a set of landmark points for the registration process. In this study, registration parameters were found by another approach on the basis of matched control points.

The other approach of using neural networks in image registration is seen in a study (Wachowiak, Smolikova, Zurada, & Elmaghraby, 2002) for landmark-based elastic biomedical image registration. In this study, local elastic registration is achieved by feedforward, Gaussian-sigmoid and radial basis function neural networks. This study also detects and matches control points. After control points are matched, simply, a local elastic model is used to interpolate the coordinates of the other points in one image to those in the other image by the help of a neural network.

In the same way support vector machines can be used in such problems. Namely, support vector machines can also establish a nonlinear transformation between two images. While establishing the nonlinear transformation between the two images, some control points from the images are used. This approach was used in (Peng, Liu, Tian, & Zheng, 2006) and (Davoodi-Bojd, & Soltanian-Zadeh, 2008).

The first image registration scheme that estimates the registration parameters by a feedforward neural network (FNN) own its own have been proposed by Elhanany (Elhanany, Sheinfeld, & Beck, 2000). The registration scheme proposed estimates the affine transformation parameters of a test image with respect to a reference image using discrete cosine transform (DCT) features as a global image feature set. In their study, the inputs of a trained neural network are selected DCT coefficients of a test image. As a result, the estimated parameters are obtained at the output. In their work, there exists a pre-registration stage in which DCT features extracted from a set of translated, rotated and scaled copies of the reference image are employed to train a FNN. This was a new method in image registration and their work pioneered a new category of registration schemes.

After this study, some other studies followed. However, they used Zernike moments, (Wu, & Xie, 2004) principal components (Xu, Jin, & Guo, 2004) and kernel independent components (Xu, Jin, Guo, & Bie, 2004) instead of DCT features.

All the registration schemes given above provide rather accurate results for noisy images. However they have two drawbacks. The first one is the duration of learning process of FNN. This period of iterative learning process of FNN is too long. The second drawback is the difficulty with network generalization for problem specific data to increase the accuracy of the results.

As it is known, in order to obtain a well generalized FNN, regularization techniques or early stopping method can be used. However, these solutions cost either increased training times or many number of training trials. When the training phase is kept too long, that results a network which is not generalized enough because of overlearning of training data used in that training phase.

The fundamental objective of this thesis is to replace the FNN with a radial basis function neural network (RBFNN) to overcome the drawbacks mentioned above. The scheme proposed here avoids the drawbacks of a FNN-based scheme. At the same time, it increases the accuracy and gives robust results in the presence of additive Gaussian noise owing to the better generalization ability of RBFNN.The registration transformation assumed in this work is global affine transformation. This transformation is composed of the Cartesian operations of a scaling, a translation, and a rotation and the aim is to find these parameters of a transformed image for which a neural network is trained.

In this thesis, a neural network estimates the affine transformation parameters of a test image with respect to a reference image using discrete cosine transform (DCT) and 2-dimensional principal component analysis (2D-PCA) features as global image feature sets respectively. Selected DCT and 2D-PCA coefficients of a test image are inputs to the trained network and estimated parameters are obtained at the output. In a pre-registration phase, features extracted from a set of translated, rotated and scaled copies of the reference image are employed to train neural networks. In this thesis, Feed-forward Neural Network (FNN) and Radial Basis Function Neural Network (RBF) are implemented to compare their performances. In the testing phase a new data set of translated, rotated and scaled copies of the reference image are employed to test the networks. The results are compared in the conclusion part.

1.2 Outline

This thesis has five chapters. Chapter 1 presents introduction to image registration, general review of the implementation that is done with artificial neural networks.

In chapter 2, image registration theory is given. Image registration methods and usage of image registration with artificial neural networks are also mentioned.

In this thesis, image registration is implemented using a neural network and, because of this; neural network based image registration is presented in chapter 3 in detail. Neural Network architectures and training algorithms are also examined.

In Chapter 4, the experimental work is given. In this thesis, image registration is realized with two different artificial neural network methods, which are Feed-forward Neural Network and Radial Basis Function Neural Network. While implementing these methods, two different types of feature data sets were extracted using Discrete Cosine Transform and Two Dimensional Principal Component Analysis. In this chapter extracting these data sets with two different methods and the train and the test processes are discussed. All experimental results are given in this chapter. The comparisons of the two neural network methods and two different feature sets are also presented.

Finally, a conclusion is given in chapter 5.

CHAPTER TWO

IMAGE REGISTRATION THEORY

2.1 Introduction to Image Registration

Sets of data acquired by sampling the same scene or object at different times, or from different perspectives, will be in different coordinate systems in computer vision. Image registration is the process of transforming such kind of different sets of data into one coordinate system. That is; image registration is the process of overlaying two or more images which belong to the same scene. These images are taken at different times, from different viewpoints, and/or by different sensors. It geometrically aligns the reference and sensed images. (Zitova & Flusser, 2003)

Typically, the application areas of registration are remote sensing (multispectral classification, environmental monitoring, change detection, image mosaicing, weather forecasting, creating super-resolution images, integrating information into geographic information systems (GIS)), medicine (combining computer tomography (CT) and NMR data to obtain more complete information about the patient, monitoring tumor growth, treatment verification, comparison of the patient’s data with anatomical atlases), cartography (map updating), and in computer vision (target localization, automatic quality control).

2.2 Theory Definition

As it was mentioned previously, image registration can be defined as a mapping between two images. (Brown, 1992). This mapping is spatially and it is with respect to intensity. If these two images are defined as two 2D arrays of a given size denoted by I1 and I2, then the mapping between images can be expressed as:

$$I_2,(x, y) = g(I_1(f(x, y))) \tag{2.1}$$

Where $I_1(x,y)$ and $I_2(x,y)$ each map to their respective intensity (or other measurement) values and *f* is a 2D spatial-coordinate transformation, and *g* is 1D intensity or radiometric transformation.

2.3 Transformations

The type of spatial transformation or mapping used to properly overlay two images is the fundamental characteristic of any image registration technique.

The most common general transformations are rigid, affine, projective, perspective, and global polynomial. These transformations and their parameters are defined in this section.

First of all, a transformation T is linear if for every constant c

$$T(x_1+x_2) = T(x_1)+T(x_2) \tag{2.2}$$

and

$$cT(x)=T(cx) \tag{2.3}$$

If the objects in the images retain their relative shape and size, rigid transformations are used in object or sensor movement in such kind of images. Briefly, a rigid-body transformation is the combination of a rotation, a translation, and a scale change.

Affine transformation is the most commonly used registration transformation. This transformation is sufficient to match two images. These images are of a scene which has the same viewing angle but taken from a different position. Affine transformation is a combination of Cartesian operations of a scaling, a translation and a rotation. A transformation is defined as affine if T(x)-T(0) is linear. It generally has four parameters. If these parameters are denoted as t_x, t_y, s and θ that map a point

(x_1,y_1) of the first image to a point *(x_2,y_2)* of the second image it is represented as follows:

$$\begin{pmatrix} x_2 \\ y_2 \end{pmatrix} = \begin{pmatrix} t_x \\ t_y \end{pmatrix} + s \times \begin{pmatrix} cos\theta & -sin\theta \\ sin\theta & cos\theta \end{pmatrix} \times \begin{pmatrix} x_1 \\ y_1 \end{pmatrix} \tag{2.4}$$

The general 2D affine transformation is:

$$\begin{pmatrix} x_2 \\ y_2 \end{pmatrix} = \begin{pmatrix} a_{13} \\ a_{23} \end{pmatrix} + s \times \begin{pmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{pmatrix} \times \begin{pmatrix} x_1 \\ y_1 \end{pmatrix} \tag{2.5}$$

Projective transformations and perspective transformations account for distortions because of the projection of objects which are at varying distances to the sensor onto the image plane. Perspective transformation is more general than the projection one in applications.

To use the perspective transformation for registration, the relative distance of the objects of the scene and the sensor is needed to be known.

If the coordinates of an object point in the scene are (x_0, y_0, z_0), then the corresponding point in the image is

$$x_i = \frac{-f x_0}{z_0 - f} \tag{2.6}$$

$$y_i = \frac{-f y_0}{z_0 - f} \tag{2.7}$$

where, *f* is the position (focal length) of the center of the camera lens when the camera is in focus for far objects.

2.4 Distortion

In a given problem, one of the important factors that has to be taken into consideration is the source of misregistration. The misregistration source is the reason of the misalignment between images. Misregistration happens because of such that change in a sensor position, viewpoint and viewing characteristics or it can be due to object movement and deformation.

Distortions can be classified as static/dynamic, internal/external and geometric/photometric.

The reason of internal distortions is sensor. On the other hand, external distortions arise from the sensor operations that continuously changed. The other reason of external distortions is individual scene characteristics.

Most of the internal errors and many of the photometric distortions are static and they can be removed by using calibration. The reasons of intensity distortions that are not static are change in sensor and varied lighting and atmospheric conditions. Predictably; in a particular system, the more that is known about the type of distortion present, the more effective registration can be.

2.5 Rectification

It can be said that, registration can be easily implemented to a system which the scene under observation is relatively flat and the viewing geometry is known. This is often the case in remote sensing if the altitude is sufficiently high. In this type of registration, rectification is applied in order to correct the perspective distortion in an image of flat scene. The effect of perspective distortion is compressing the image of scene features as if they are farther from the camera. Rectification is used to correct images so that they conform to a specific map standard such as Universal Transverse Mectacor projection. It is also used in registering two images of a flat surface that were taken from different viewpoints.

2.6 Image Registration Methodology

As it was mentioned above, image registration is widely used in remote sensing, medical imaging, computer vision etc. In general, according to the manner of the image acquisition image registration applications can be divided into four main groups:

Different viewpoints (multiview analysis) method is related to the images of the same scene which are acquired from different viewpoints. The objective of this method is to gain larger a 2D view or a 3D representation of the scanned scene.

Different times (multitemporal analysis) method is related to the images of the same scene that are acquired at different times, often on regular basis, and possibly under different conditions. The aim of this method is finding and evaluating changes in the scene that appeared between the consecutive image acquisitions.

Different sensors (multimodal analysis) method is related to different sensors that acquire images of the same scene. The aim of this method is to integrate the information obtained from different sources of images in order to gain more complex and detailed scene representation.

Scene to model registration method is used for the images of the scene and of the model of the scene to align one with another. As an example; the model can be a computer representation of the scene maps or digital elevation models (DEM) in GIS, another scene with similar content (another patient), ‘average’ specimen, etc. The aim of this method is to localize the acquired image in the scene/model and/or to compare them.

2.7 The steps of Image Registration

2.7.1 Feature detection

The objects which have the properties such that closed-boundary regions, edges, contour, line intersections, corners are manually or, preferably, automatically detected. In order to process further, these features can be represented by their point representatives which can be centers of gravity, line endings and distinctive points. These point representatives are called control points (CPs) in the literature.

2.7.2 Feature matching

A relationship between the features detected in the sensed image and that are detected in the reference image is established in this step. To this end; various feature descriptors and similarity measures along with spatial relationships among the features are used.

2.7.3 Transform model estimation

In this step, the type and parameters f the mapping functions which align the reference and test images are estimated. The established feature correspondence is used to compute the parameters of the mapping functions.

2.7.4 Image resampling and transformation

The mapping functions are used to transform the sensed image. The image values which have non-integer coordinates are computed by a suitable interpolation technique.

In this step, for a given task at first we have to decide what kind of features are appropriate. This means that with the help of which features we detect the images. The most important property of the features used is their distinctivity. That is they have to be distinctive objects which are frequently spread over the images and which are easily detectable. In the ideal case, the algorithm used should be able to detect the same features in all of the projections of the scene.

In an ideal case, the algorithm should be capable of detecting the same features in all projections of the scene regardless of the particular image deformation.
Problems can arise in the feature matching step. These problems can arise because of incorrect feature detection or by image degradations.

CHAPTER THREE

NEURAL NETWORK BASED IMAGE REGISTRATION

3.1 Introduction to Neural Network Based Image Registration

In a typical neural network based image registration scheme, there are two separate phases. These phases are shown in Figure 3.1. In the pre registration phase, in order to generate a set of affine transformed parameters, a reference image is rotated, scaled and translated several times. After adding noise to these images, in order to obtain the training data for neural network, a global feature extraction is applied to every image in the set. As examples of global features, DCT coefficients and moments of images can be given.

In the training stage, after extracting the global features from the image set, these features are fed into a neural network together with corresponding parameter values at the output. After the trained network is ready, registration phase is simple. That is; the same global features are extracted from a test image with unknown affine transformation parameters. These features are fed to the network, and then the estimated parameter values are read at the output.

In this type of approach of registration, the entire registration problem is reduced to vector regression by a neural network. The neural network used in this situation provides an accurate mapping between global feature space of affine transformed images and their affine transformation parameters which are known as registration parameters. The advantage of this method for the applications which require many fast registrations to a single reference image is good mapping capability and ease of use of neural networks. Some application areas of this method are image-based navigation, super-resolution, and analysis of time series of medical images.

In order to improve the generalization and immunity of the neural network, noise is added to the images in the training set. The number of affine transformed images represents the samples from the four-dimensional parameter space. This number must be sufficient and the sampling of the parameter space must be done suitably. Else, the parameter estimation accuracy of the network will not be satisfactory.

In the pre-registration phase, when we use an FNN, the training stage is lengthy. Moreover, in the registration phase, the output accuracy of the scheme with FNN strongly depends on how good the FNN has been trained. On the other hand, if the FNN is replaced with a RBFNN this change simplifies the training stage both in terms of training time and improving network generalization.

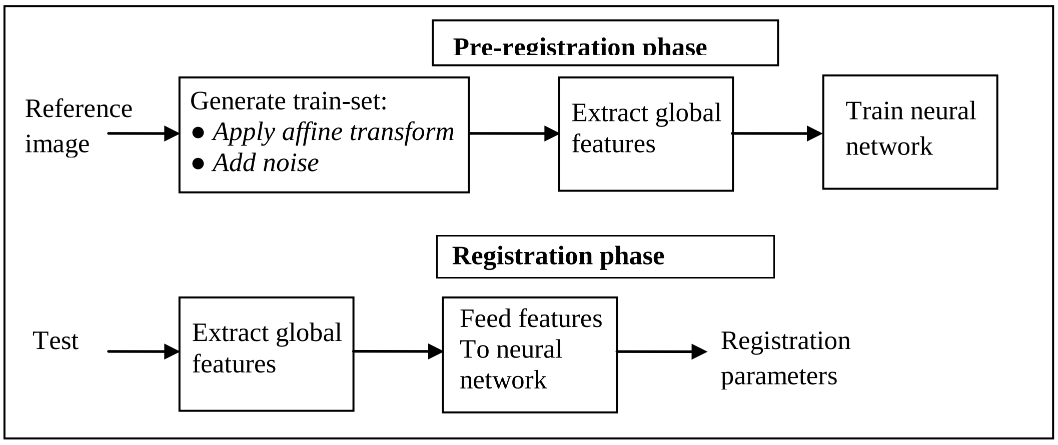

Figure 3. 1 Neural network based registration scheme

3.1.1 General Review of Neural Networks

A neural network is called so because it is a network of interconnected elements that behaves like neurons. It can be said that neural network is an attempt to realize biological neuron model to the machines that are desired to work in a similar way to the human brain. It can be defined as a massively parallel distributed processor that has a natural propensity for storing experiential knowledge and making it available for use.

It has resemblance to the brain in two respects:

1. The network acquires knowledge through a learning process

2. The synaptic weights which also can be referred as interconnection strengths are used to store the knowledge.

The function of a neural network is producing an output pattern when an input pattern is presented. To summarize, a neural network should be able to:

- classify patterns
- be small enough to be physically realizable
- be programmed by training, so it must have the ability to learn
- be able to generalize from the examples shown during training.

Neural network are composed of computation elements that works parallel. These elements were designed originating from biological neural systems. In order to realize a function, we can train e neural network by adjusting the values of the connections that are called weights.

In general; by training the neural networks the desired outputs can be obtained. This is given in Figure 3.1. Until the output of the neural network reaches the desired target, the output and the target are compared and training process continues.

Neural networks are used in the areas of pattern recognition, identification, classification, speech processing, image processing with computer and control systems, etc.

In general, the learning process may be classified as follows:

• Learning with a teacher, also referred as supervised learning.

• Learning without a teacher, also referred as unsupervised learning.

In this thesis, we only deal with the methods of Feedforward Neural Network and Radial Basis Function neural network. These methods are both related to the supervised learning. Therefore we will only deal with these two methods in theory.

3.2 Neuron Model

A simple model of artificial neuron is a computation element that has one output and more than one input. The model of single-input neuron is given in the Figure 3.2 referred to the study (Haykin, 1999).

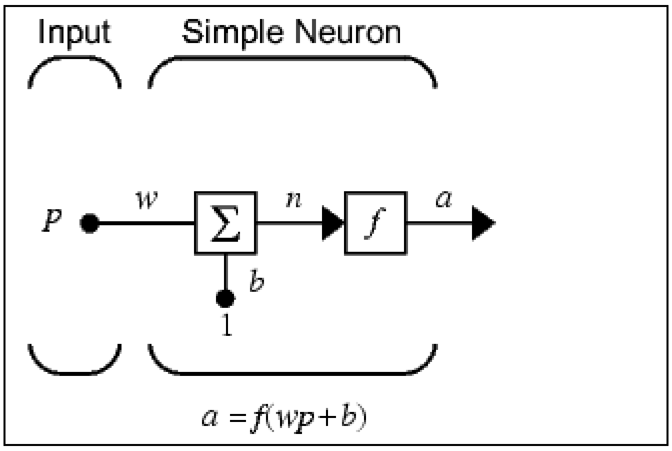

Figure 3. 2 Model of a single input neuron

In Figure 3.2; p is the scalar value which refers to the input of the neuron, w is the weight value between the neuron and the p value, b is the bias value of the neuron, f is transfer function, a is the output of the neuron. At first; input p is multiplied by the weight w. The result of this operation is added to the bias b and after the operation with the transfer function, the output a is calculated. Transfer function can be a linear or non-linear differential function. In this simple neuron model, there is a simple mathematical relationship between input and output. However, in order to the neuron gives the desired output value, w and b values must be adjusted to an optimum value. The neuron model of multi-inputs is given in the Figure 3.3 referred to the study (Haykin, 1999).

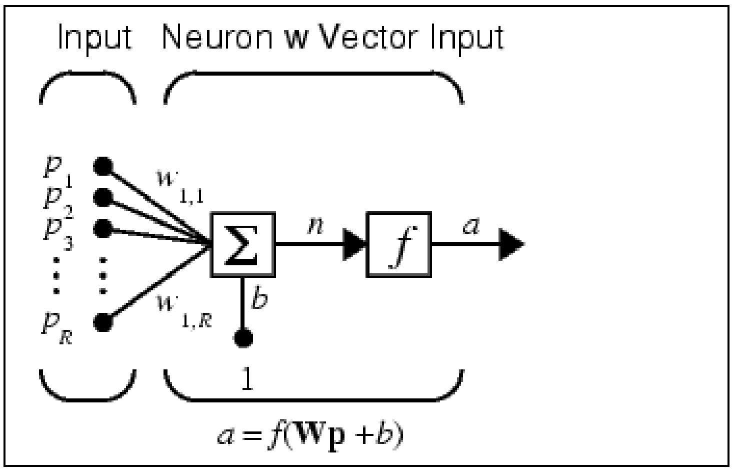

Figure 3. 3 Model of multi-inputs neuron

In this figure, R is the number of inputs. As it can be seen from the figure the number of weight values w is equal to the number of input values. In this case; the mathematical expression of the neuron is given below:

$$a = f\left(\sum_{i=1}^{R} w_i p_i + b\right) \tag{3.1}$$

3.3 Network Architectures

The structure of neural network is directly related to the learning algorithm used. In general, there are three different kinds of network architectures

3.3.1 Single Layer Feedforward Networks

Neurons are organized in the form of layers in a layered neural network. The simplest form of a layered network has one input layer, and an output layer. This network is shown in Figure 3.4

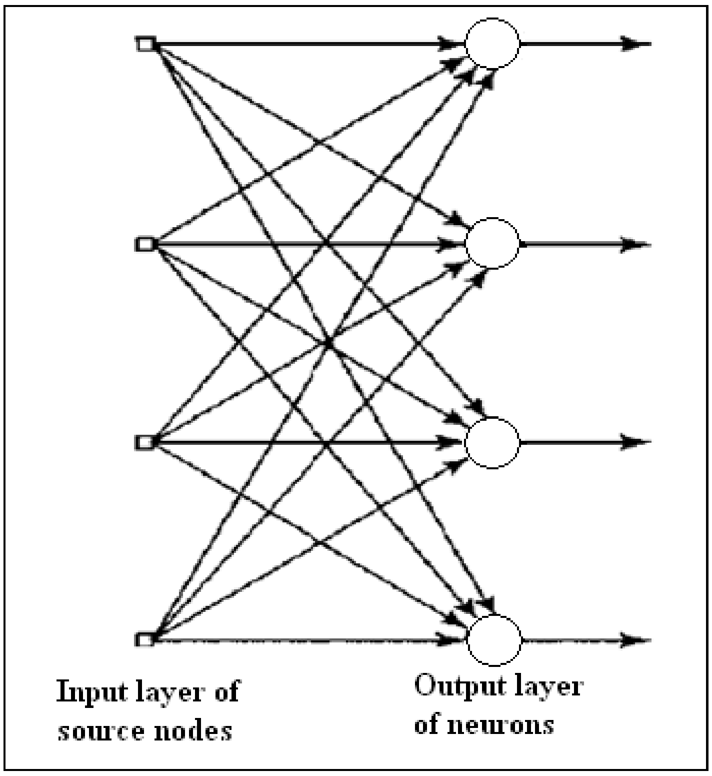

Figure 3.4 Single-layered feedforward neural network

3.3.2 Multilayer Feedforward Networks

In this kind of neural network in addition to the input and output layers, there are also one or more hidden layers whose computation nodes are correspondingly called hidden neurons or hidden units. These hidden layers are between input and output layers. The outputs of the input nodes are input to the first hidden layer, the outputs of the first hidden layer are inputs to the second hidden layer and so on. Figure 3.5 shows a one hidden layered feed-forward neural network.

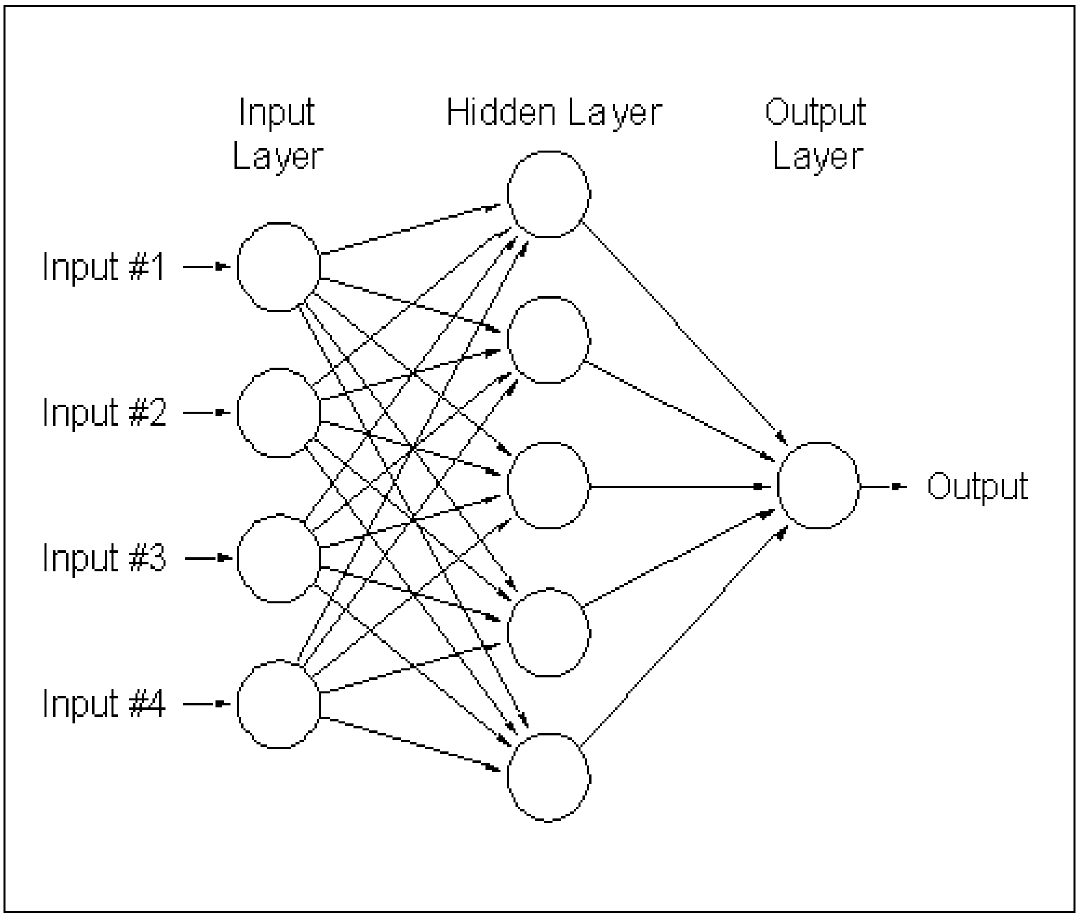

Figure 3. 5 A multilayer feedforward neural network with one hidden layer

3.3.3 Recurrent Networks

A recurrent neural network is distinguished from a feed-forward neural network by at least one feedback loop it has. The presence of feedback loops in recurrent neural network has a profound effect on learning capability of the network.

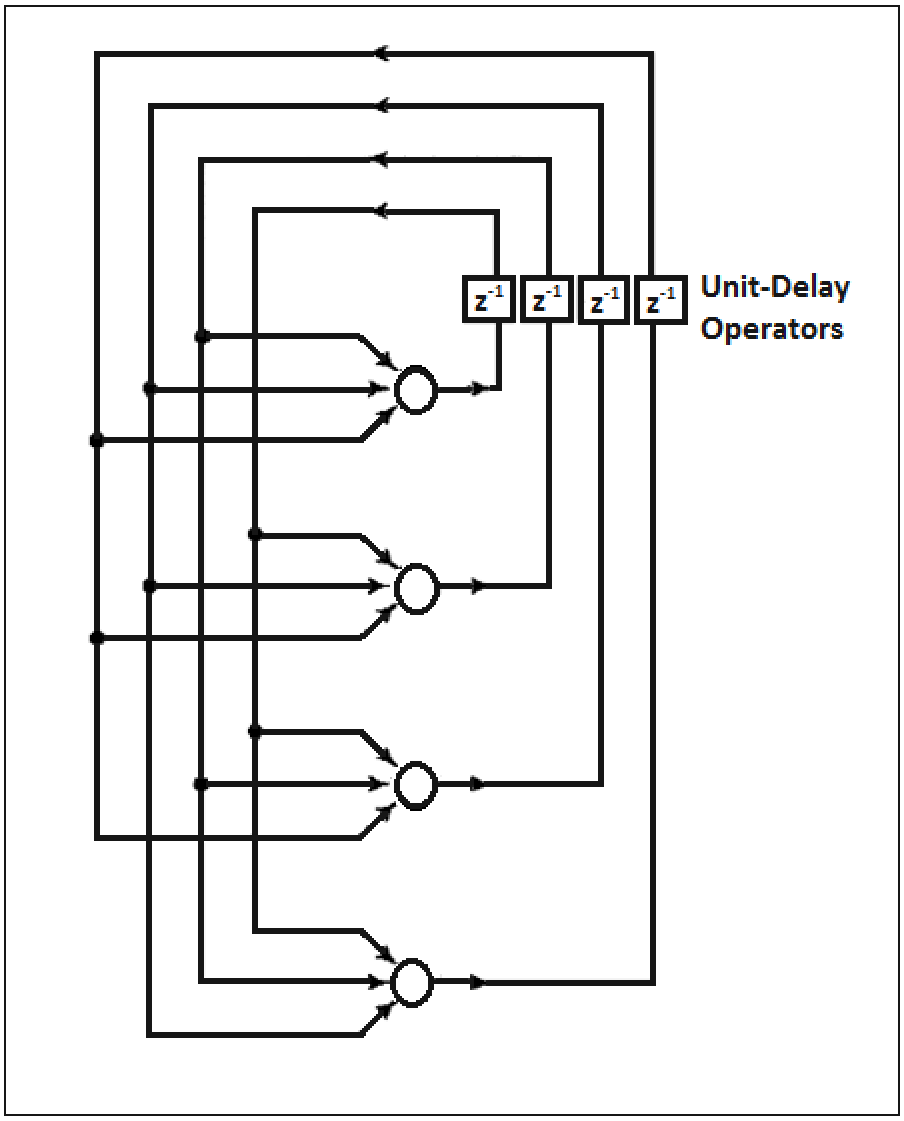

Figure 3.6 Recurrent Neural Networks with no self feedback loops and no hidden neurons

3.4 Transfer Functions

3.4.1 Hard-Limiter Transfer Function

The graph of hard limiter transfer function is given in Figure 3.7 below. In this graph, n is the input of the function and a is the output of the function. The formula of this function is a=f(n). In this function; if the input value a is greater than zero, the output value is one; if the input value is smaller than zero, the output value is zero. Generally, this function is used in classification applications.

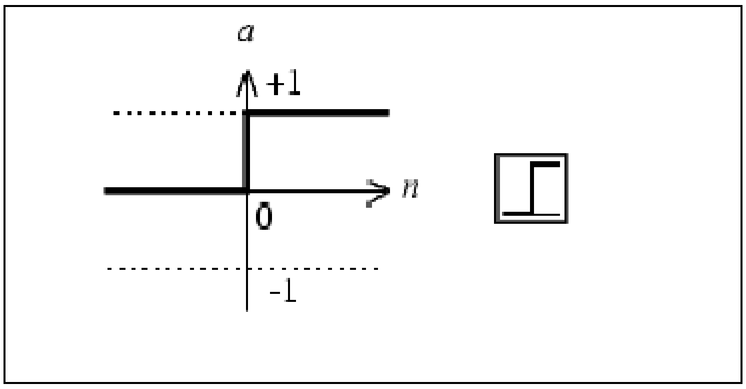

Figure 3.7 Hard Limit Transfer Function

3.4.2 Linear Transfer Function

The graph of linear transfer function is given in the Figure 3.8. As it can be seen from the figure, input is given to the output without any change. Here *a* is equal to *n*.This function is commonly used in linear filter problems.

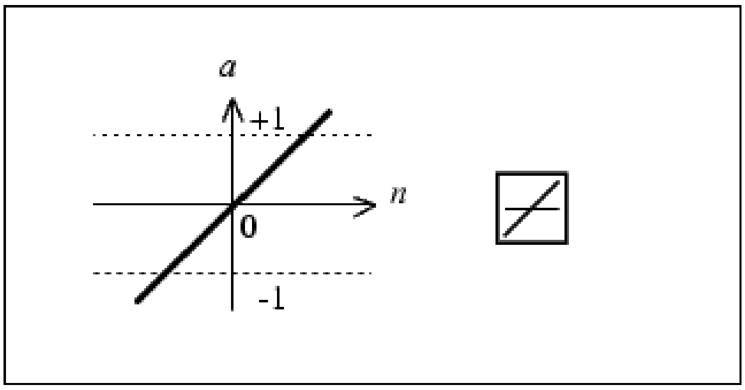

Figure 3.8 Linear Transfer Function

3.4.3 Log-Sigmoid Transfer Function

The sigmoid is a nonlinear logarithmic function. The output values are between zero and one independent of in what interval the input values are. It is a differential function, so it can be used with back-propagation algorithms. It can be used in the solution of nonlinear problems. The graph of log-sigmoid transfer function is shown in Figure 3.9.

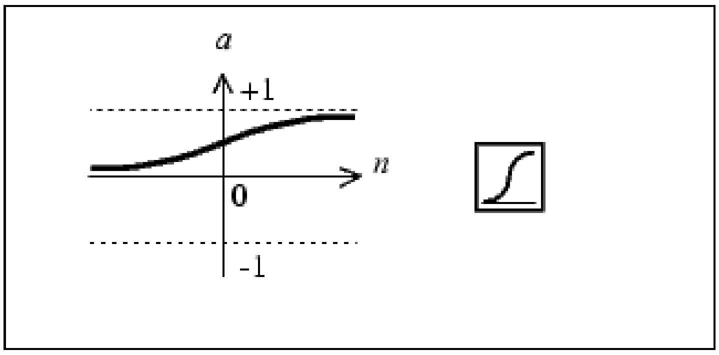

Figure 3.9 Log-sigmoid transfer function.

The mathematical expression of the function is as equation 3.2 below:

$$a = logsig(n) = \frac{1}{(1+10^{(-n)})} \tag{3.2}$$

3.4.4 Tan-Sigmoid Transfer Function

The graph of tangent sigmoid transfer function is given in the Figure 3.10. The mathematical expression of tan-sigmoid transfer function is as equation 3.3 below:

$$a = tansig(n) = \frac{2}{(1+10^{(-2\times n)})} - 1 \tag{3.3}$$

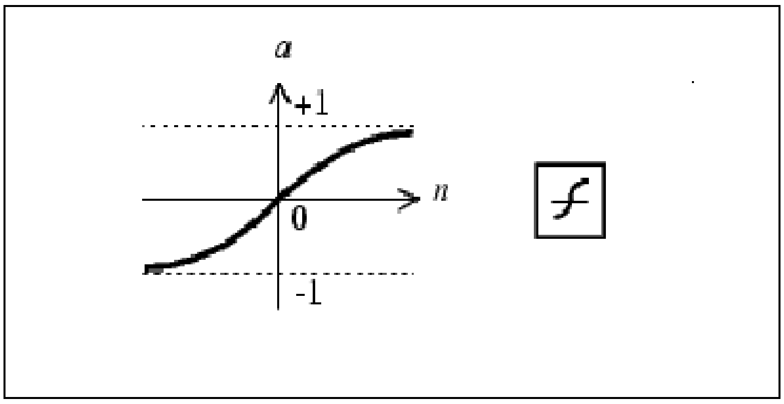

Figure 3.10 Tan-sigmoid transfer function

3.5 Feed-Forward Neural Network

A feedforward neural network has connections between units that do not form a directed cycle. In this type of network, the information moves only in forward direction. The information moves from the input nodes through the hidden nodes and to the output nodes. In the network there are no cycles or loops.

3.5.1 Single Layer Perceptron

The simplest kind of feedforward neural network is a *single-layer perceptron* network. This type of neural network consists of a single layer of output nodes and the inputs are fed directly to the outputs via a series of weights. By taking this architecture into consideration, this type of neural network can be denoted as the simplest kind of feed-forward network. In each node the sum of products of the weights and the inputs is calculated. If the calculated value in this multiplication is above some threshold the neuron fires and takes the activated value; otherwise it takes the deactivated value. Generally, the threshold value is 0, activated value is 1 and deactivated value is -1. The neurons which have such kind of activation functions are also called artificial neurons or linear threshold units. . In the literature the term perceptron often refers to networks consisting of just one of these units. As long as the threshold values lies between the activated and deactivated states, a perceptron can be created using any values of activated and deactivated states.

A simple learning algorithm called delta rule is used for training perceptrons. This algorithm calculates the errors between calculated output and sample output data, and uses the calculated result to create an adjustment to the weights. It can be said that delta rule is a kind of implementation of gradient descent.

Single-unit perceptrons are only capable of learning linearly separable patterns. In order to learn more complicated patterns, More complicated patterns can be learned by multilayer perceptrons.

3.5.2 Multilayer Perceptrons and Back-Propagation Learning

The back-propagation algorithm has an important role in the design of a special class of layered feedforward networks known as multilayer perceptrons (MLP). Multilayer perceptron consists of an input layer of source nodes and an output layer of neurons known as computation nodes as shown in Fig.3.11. With the help of these two layers the network is connected to the outside world. In addition to these two layers, the multilayer perceptron usually has one or more layers of hidden neurons. These neurons are called as hidden because they are not directly accessible. The hidden neurons extract important features contained in the input data.

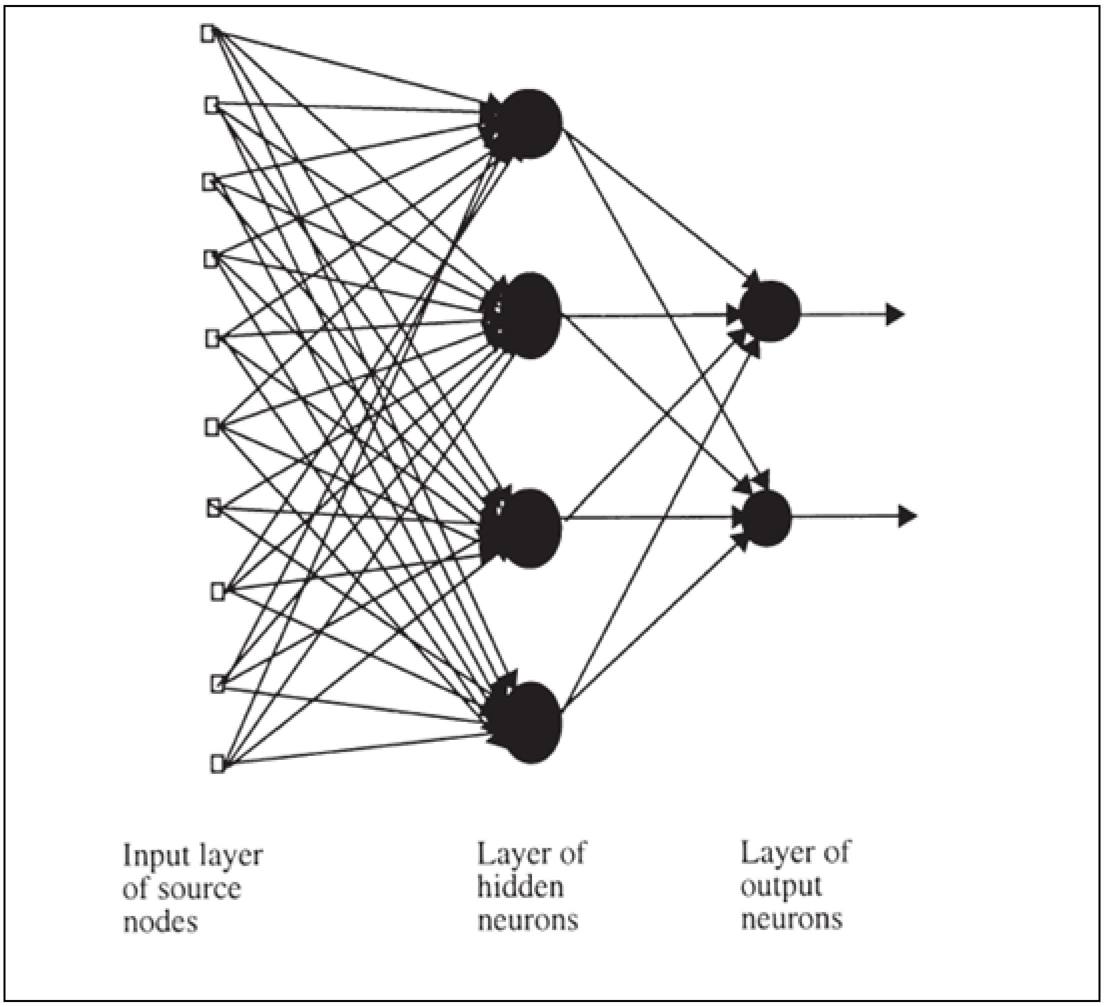

Figure 3.11 Fully connected feed-forward with one hidden layer and one output layer

MLP is usually trained by using a backpropagation (BP) algorithm that involves two phases:

• Forward Phase: The free parameters of the network are fixed during this phase, and the input signal is propagated through the network seen in Figure 3.11 layer by layer. When an error signal is computed, the forward phase finishes

$$e_i = d_i - y_i \tag{3.4}$$

In the equation 3.4, di is the desired response and yi is the actual output produced by the network in response to the input xi.

• Backward Phase During this second phase known as backward phase; the error signal e_i is propagated through the network in the backward direction. During this phase in order to minimize the error e_i some adjustments are applied to the free parameters of the network.

Back-propagation learning may be implemented in one of two basic ways:

1. Sequential mode: This mode is also referred to as the on-line mode or stochastic mode. In this mode, adjustments are made to the free parameters of the network on an example basis. The sequential mode is suitable for pattern classification.

2. Batch mode: In this second mode of BP learning, adjustments are made to the free parameters of the network on an epoch basis. Each epoch consists of the entire set of training examples. The batch mode is suitable for nonlinear regression. The advantage of the back-propagation learning algorithm is its ease of implementation and computational efficiency. However, a major limitation of the algorithm is when we have to deal with a difficult learning task that requires the use of a large network, this type of network does not always converge and can be excruciatingly slow.

3.5.3 Accelerated Learning Backpropagation Algorithms

The summary of accelerated learning backpropagation algorithms is given in the next parts with reference to the study (Ham & Kostanic, 2001, chap. 3).

3.5.3.1 Conjugate –Gradient-based algorithm for training an MLP

Step 1: Inıtialize the network weights to some small random values.

Step 2: Propagate the *q*th training pattern through the network, calculating the output of every node.

Step 3: Calculate the local error at every node in the network. For the output nodes the local error is calculated as

$$\delta_{i,q}^{(s)} = \left(d_{i,q} - x_{out,i,q}^{(s)}\right) g(v_{i,q}^{(s)}) \tag{3.5}$$

Where g(.) is the derivative of activation function f(.). For each of the hidden layer nodes, the local error is calculated as

$$\delta_{i,q}^{(s)} = \left(\sum_{h=1}^{n_{s+1}} \delta_{h,q}^{(s+1)} \, w_{h,i}^{(s+1)}\right) g(v_{i,q}^{(s)}) \tag{3.6}$$

Step 4: For each of the linear combiner estimates, the desired output value is given by

$$\tilde{v}_{i,q}^{(s)} = f^{-1}(\bar{d}_{i,q}^{(s)}) \text{ where } \bar{d}_{i,q}^{(s)} = x_{out,i,q}^{(s)} + \mu\delta_{i,q}^{(s)} \tag{3.7}$$

Step 5: Update the estimate of the covariance matrix in each layer

$$C^{(s)}(k) = bC^{(s)}(k-1) + x_{out,q}^{(s-1)} x_{out,q}^{(s-1)T} \tag{3.8}$$

Update the estimate of the cross-correlation vector for each node

$$p_i^{(s)}(k) = bp_i^{(s)}(k-1) + \tilde{v}_i^{(s)} x_{out,q}^{(s-1)} \qquad (3.9)$$

Where k is the pattern presentation index.

Step 6: Update the weight vector for every node in the network as follows.

a) At every node calculate

$$g_i^{(s)}(k) = C^{(s)}(k) w_i^{(s)}(k) - p_i^{(s)}(k) \text{ , else} \qquad (3.10)$$

If $g_i^{(s)}$ =0, do not update the weight vector for the node and go to step 7, else perform the following steps:

b) Find the direction d(k). If the iteration number is an integer multiple of weights in the node, then

$$d_i^{(s)}(k) = -g_i^{(s)}(k) \qquad (3.11)$$

Else

$$d_i^{(s)}(k) = -g_i^{(s)}(k) + \beta_i^{(s)} d_i^{(s)}(k-1) \qquad (3.12)$$

Where

$$\beta_i^{(s)} = -g_i^{(s)T}(k) \frac{C^{(s)}(k) d_i^{(s)}(k-1)}{d_i^{(s)T}(k-1) C^{(s)}(k) d_i^{(s)}(k-1)} \qquad (3.13)$$

c) Compute the step size

$$\propto_i^{(s)}(k) = -\frac{g_i^{(s)}(k)d_i^{(s)}(k)}{d_i^{(s)T}(k)C^{(s)}(k)d_i^{(s)}(k)} \tag{3.14}$$

d) Modify the weight vector according to

$$w_i^{(s)}(k) = w_i^{(s)}(k-1) + \propto_i^{(s)}(k)d_i^{(s)}(k) \tag{3.15}$$

Step 7: If the network has not converged, go back to step 2.

3.5.3.2 Recursive Least-Squares-Based Backpropagation Algorithm

Step 1: Inıtialize the network weights to some small random values.

Step 2: Present an input pattern and calculate the responses of all linear combiners $v_i^{(s)}$ and all neuron outputs $x_{out,I}^{(s)}$ in the network.

Step 3: For each layer of the network, calculate the Kalman gain matrix and update the covariance matrix estimate according to the following equations:

$$A^{(s)}(k) = \left[C^{(s)}(k-1)\right]^{-1} x_{out}^{(s-1)}(k) \tag{3.16}$$

Update the Kalman gain matrix for the *s*th layer.

$$K^{(s)}(k) = \frac{A^{(s)}(k)}{b + x_{out}^{(s-1)T}(k)A^{(s)}(k)} \tag{3.17}$$

Update the covariance matrix for the sth layer according to

$$\left[C^{(s)}(k)\right]^{-1} = b^{-1}\left\{\left[C^{(s)}(k-1)\right]^{-1} - K^{(s)}(k)A^{(s)T}(k)\right\} \tag{3.18}$$

Step 4: Calculate and backpropagate the local errors for the output layer according to the equations:

$$\boldsymbol{\delta}_{\boldsymbol{i}}^{(\boldsymbol{s})} = (\boldsymbol{d}_{\boldsymbol{qh}} - \boldsymbol{x}_{\boldsymbol{out,i}}^{(\boldsymbol{s})})\boldsymbol{g}(\boldsymbol{v}_{\boldsymbol{i}}^{(\boldsymbol{s})}) \tag{3.19}$$

And for hidden layers

$$\boldsymbol{\delta}_{\boldsymbol{i}}^{(\boldsymbol{s})} = \left(\sum_{h=1}^{n_2} \boldsymbol{\delta}_{\boldsymbol{h}}^{(\boldsymbol{s+1})}\, w_{hi}^{(s+1)}\right) g(v_i^{(s)}) \tag{3.20}$$

Where g(z)=df(z)/dz, and f(z) is the activation function of the neuron.

Step 5: For each of the linear combiners, estimate the desired output according to

$$\tilde{v}_{i,q}^{(s)} = f^{-1}(\boldsymbol{x}_{\boldsymbol{out,i}}^{(\boldsymbol{s})} + \mu \boldsymbol{\delta}_{\boldsymbol{i}}^{(\boldsymbol{s})}) \tag{3.21}$$

Where $f^{-1}(z)$ denotes the inverse of the neuron activation function.

Step 6: Update the weights in each layer of the network according to

$$w_i^{(s)}(k) = w_i^{(s)}(k-1) + K(k)[\tilde{v}_i^{(s)}(k) - v_i^{(s)}(k)] \tag{3.22}$$

Step 7: Stop if the network has converged, else go back to step 2.

3.5.3.3 Backpropagation algorithm with adaptive Slopes of Activation functions

Step 1: Inıtialize the network weights to some small random values.

Step 2: From the set of the training input/output pairs, present the input pattern and calculate the network represented.

Step 3: Compare the desired network response with the actual output of the network, and the local errors are computed using

$$\delta_i^{(s)} = \left(d_{qh} - x_{out,i}^{(s)}\right) g(v_i^{(s)}) \quad \text{for output layer} \tag{3.23}$$

$$\boldsymbol{\delta}_{\boldsymbol{i}}^{(\boldsymbol{s})} = \left(\sum_{h=1}^{n_2} \boldsymbol{\delta}_{\boldsymbol{h}}^{(\boldsymbol{s+1})} w_{hi}^{(s+1)}\right) g(v_i^{(s)}) \text{ for hidden layers} \tag{3.24}$$

Step 4: The weights of the network are updated according to

$$w_{ij}^{(s)}(k+1) = w_{ij}^{(s)}(k) + \mu^{(s)} \delta_i^{(s)} x_{out,j}^{(s)} \tag{3.25}$$

Step 5: The slopes of the activation functions are updated according to

$$\gamma_i^{(s)}(k+1) = \gamma_i^{(s)}(k) + \beta \boldsymbol{\delta}_{\boldsymbol{i}}^{(\boldsymbol{s})} + \boldsymbol{\rho}\left[\gamma_i^{(s)}(k) - \gamma_i^{(s)}(k-1)\right] \tag{3.26}$$

If $\gamma_i^{(s)}(k+1) < \gamma_{min}$, then $\gamma_i^{(s)}(k+1) = \gamma_{min}$ (3.27)

Step 6: Stop if the network has converged; else go back to step 2.

3.5.3.4 Levenberg-Marquart Algorithm

Step 1: Inıtialize the network weights to some small random values.

Step 2: Present the input pattern and calculate the network represented.

Step 3: Use the formula below to calculate the elements of the Jacobian matrix associated with the input/output pairs

$$J_{i,j} \approx \frac{\Delta e_i}{\Delta w_j} \tag{3.28}$$

Step 4: When the last input/output pair is presented, use the formula below to update the weights.

$$w(k+1) = w(k) - [J_k^T J_k + \mu_k I]^{-1} J_k^T e_k \tag{3.29}$$

Step 5: Stop if the network has converged; else, go back to step 2.

3.6 Radial-Basis Function Networks

The other commonly used layered feed-forward network is the radial-basis function network which has important universal approximation properties. The structure of RBNNN is shown in Figure 3.12. With their structural simplicity and training efficiency; are good candidate to perform a nonlinear mapping between the input and output vector spaces RBF networks use memory-based learning for their design. Specifically, in this type of neural network learning is viewed as a curve-fitting problem in high-dimensional space.

RBFNN is a fully connected feed-forward structure. This type of network consists of three layers namely, an input layer, a single layer of nonlinear processing units, and an output layer. Input layer has input nodes which are equal to the dimension of the input vector x. The calculation of the output of the jth hidden neuron with Gaussian transfer function is given as

$$h_j = e^{-\left\| x - c_j \right\|^2 / \sigma^2} \tag{3.30}$$

where h_j is the output of the jth neuron, $x \in \Re^{n\times 1}$ is an input vector, $c_j \in \Re^{n\times 1}$ is the jth RBF center, σ is the center spread parameter which controls the width of the RBF, and $\left\| \cdot \right\|^2$ represents the Euclidean norm. The output of any neuron at the output layer of RBFNN is calculated as

$$y_i = \sum_{j=1}^{k} w_{ij} h_j \tag{3.31}$$

In this formula w_{ij} is the weight connecting hidden neuron j to output neuron i and k is the number of hidden layer neurons.

The mapping properties of the RBFNN can be modified by adjusting the weights in the output layer, the centers of the RBFs, and spread parameter of the Gaussian function. The simplest form of RBFNN training can be obtained with fixed number of centers. In Figure 3.12, a model of RBFNN is given.

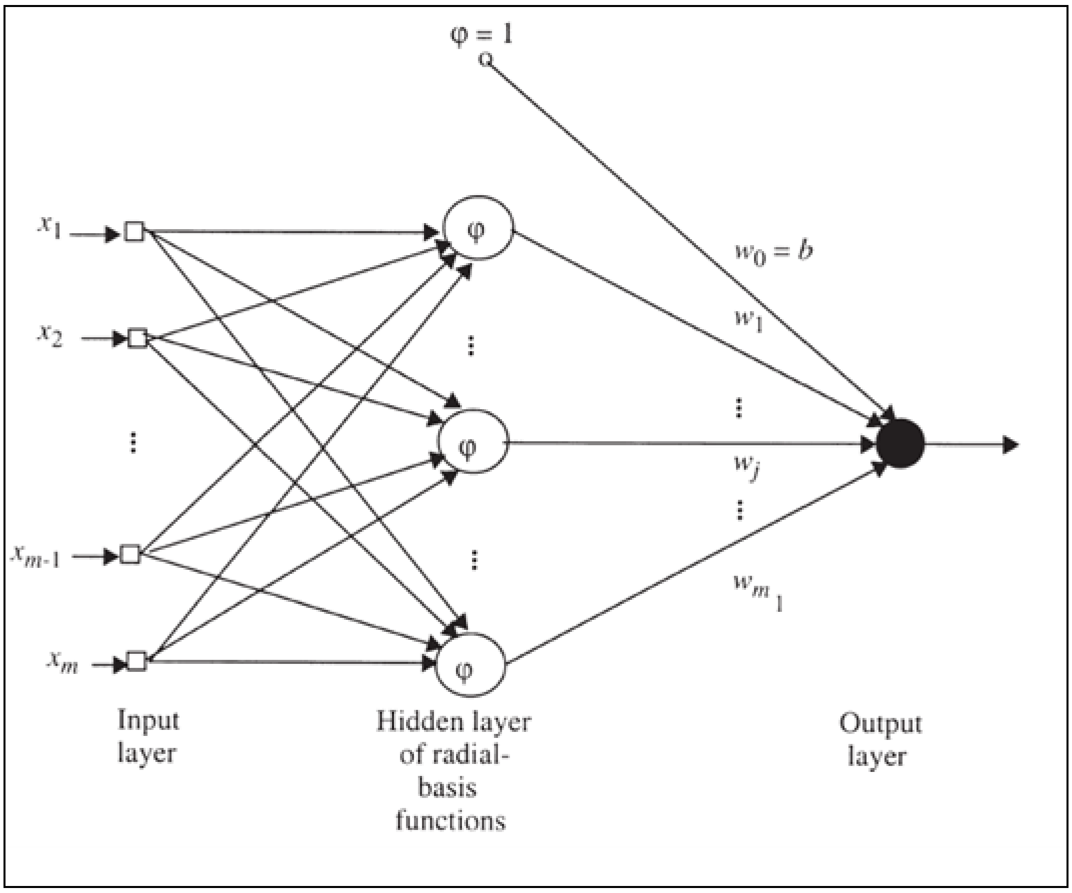

Figure 3.12 A model of RBF NN

3.7 Comparison of RBF Networks and Multilayer Perceptrons

RBFNN and multilayer perceptrons NN has differences in some fundamental respects:

RBF networks are local approximators, however multilayer perceptrons are global approximators.

RBF networks have a single hidden layer, whereas multilayer perceptrons can have any number of hidden layers.

The output layer of a RBF network is always linear, but the output layer of multilayer perceptron can be linear or nonlinear.

In an RBF network the activation function of the hidden layer computes the Euclidean distance between the input signal vector and parameter vector of the network, but the activation function of a multilayer perceptron computes the inner product between the input signal vector and the pertinent synaptic weight vector.

CHAPTER FOUR

EXPERIMENTAL WORK

4.1 General Information

In this thesis, accurate and robust image registration based on neural networks have been implemented and tested in the presence of noise. Although some studies related to image registration based on neural networks are mentioned in chapter two, section 2.7, all of the schemes given have drawbacks. These are the long period of iterative learning process of FNN and its generalization for problem specific data to increase the accuracy of the results.

As it is known; a well-generalized FNN can be obtained by regularization techniques or early stopping method at the cost of increased training times (or number of training trials). This means a lengthy training phase, that may consists of many trials for ensuring a well-generalized FNN can never be avoided.

The fundamental objective of this work is to replace the FNN with a radial basis function neural network (RBFNN) to overcome the drawbacks mentioned above. The proposed scheme here does not only avoid the drawbacks of a FNN-based scheme but also increases the accuracy and gives robust results in the presence of additive Gaussian noise owing to the better generalization ability of RBFNN.

4.2 Coarse and Fine Registration

In this thesis, the registration process is implemented in two categories. The first category is local range fine registration (LRFR) and the second category is medium range coarse registration (MRCR).

The purpose of LRFR part was to accurately estimate the registration parameters from a relatively small range using a moderate size RBFNN. If it was wanted to be accurately estimated the registration parameters from a much wider range, then the

RBFNN needed would be enormous and impractical since the number of neurons in the hidden layer of an RBFNN is equal to the number of input vectors.

On the other hand, by using MRCR, from a medium range, an additional RBFNN with a moderate size can be designed and trained to find registration parameters roughly. By this way, if it is wanted to register an image accurately from a wider parameter range: first, coarsely register image by the MRCR network, then apply back affine transform to image, and finally accurately register by the LRFR network. Later, the actual registration parameters can be found by joining the estimated values given by the two networks. The joint registration parameters are extracted from a combined affine transformation matrix. This transformation matrix is simply computed by multiplying the coarse transformation matrix from MRCR result by the fine transformation matrix from LRFR result. The same idea can also be applied to a FNN-based image registration scheme.

4.3 Data Sets

Table 4.1 shows the values of affine transformation parameters used for experimental LRFR and MRCR categories. This table also gives the numbers of generated images for training and test data. In MRCR, both range and step size chosen for all parameter values are wider compared to those in LRFR. Transformation parameters for the test data were chosen as the midpoints of the transformation parameters used for training data. This parameter choice helps discriminating the performances of the RBFNN-based scheme and FNN-based scheme by testing them at points farthest from the points at which they trained. By this way the system gives most accurate results anyway. In the experiments, three different reference images were used. In Figure 4.1 the first reference image, aerial and two samples in a training set generated from this by translating, rotating and scaling according to the values given in Table 1 is shown. Reference image moon is shown in Figure 4.2 and reference image girl is shown is Figure 4.3. All reference images have 256 gray levels and a size of 400×400 pixels. All generated images, each of which is 128 by 128 pixels size, were first added with white Gaussian noise.

In order to get DCT features, DCT of the noisy images were taken to obtain frequency domain coefficients. A region of 6 by 6 coefficients in the lowest frequency band in the DCT plane was cut out and used as a feature vector for each affine transformed image. Discarding the zero frequency coefficients, a matrix of 35 by N coefficients was obtained. In this matrix, the dimension N is the number of generated images. This matrix was used to train exact RBFNNs and FNNs. Some test images with the transformation parameter values given in Table 1 were also created and added with noise of the same strength as that in the training set. In the same way, features from the test images were obtained.

Table 4.1.Transform parameters for LRFR and MRCR

transform parameter	LRFR		MRCR	
	values used in training set	**values used in test set**	**values used in training set**	**values used in test set**
scaling	0.9, 0.965, 1.035, 1.1	0.93, 1, 1.07	0.70, 0.85, 1, 1.15, 1.3	0.77, 0.92, 1.07, 1.24
rotation (degrees)	-5, -2, 2, 5	-3, 1, 4	-30, -15, 0, 15, 30	-25, -12, 5, 22
vertical translation (pixels)	-5, -2, 2, 5	-4, 0, 3	-20, -10, 0, 10, 20	-18, -5, 7, 16
horizontal translation (pixels)	-5, -2, 2, 5	-3, 1, 4	-20, -10, 0, 10, 20	-16, -4, 4, 17
Number of generated images, *N*	256	81	625	256

For 2D-PCA coefficients, 2 Dimensional PCA is taken. As a result, a 6 by 6 matrix was obtained and this matrix was used as a feature vector for each affine transformed image. This time, we have 36 by N coefficients differently from DCT features. Same as the DCT featured system these features are used for training in both RBFNNs and FNNs. Some test images with the transformation parameter values given in Table 4.1 were also created. These created images are added with noise of the same strength as that in the training set. Features from the test images were obtained exactly in the same manner as explained for the training data.

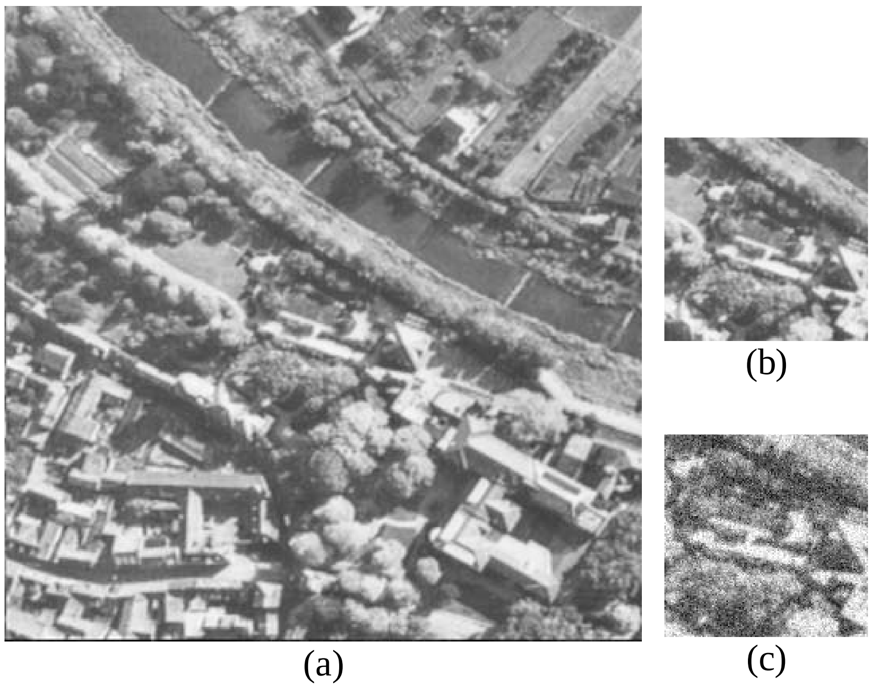

Figure 4.1 (a) Reference image *aerial*, (b) an image generated from (a) by scaling and translating and (c) another image generated by scaling, rotating, translating and adding noise to use in the training set of the neural networks.

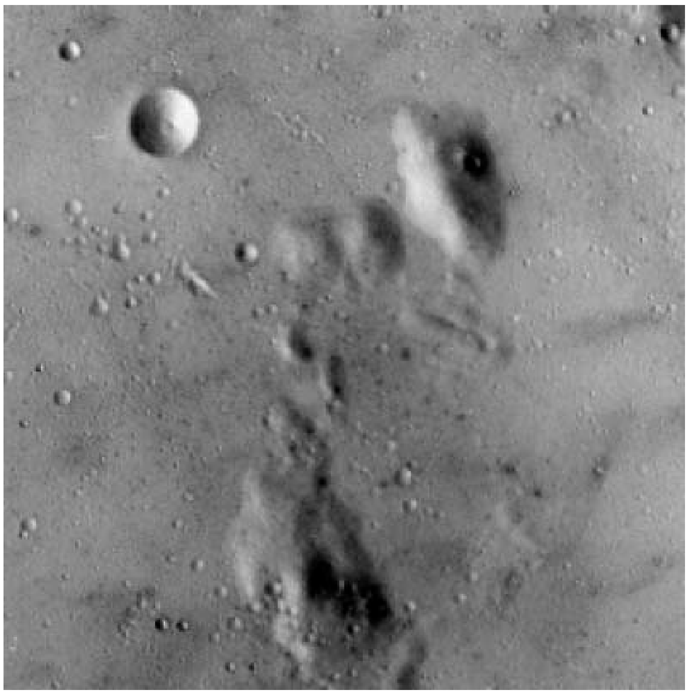

Figure 4.2 Reference image moon.

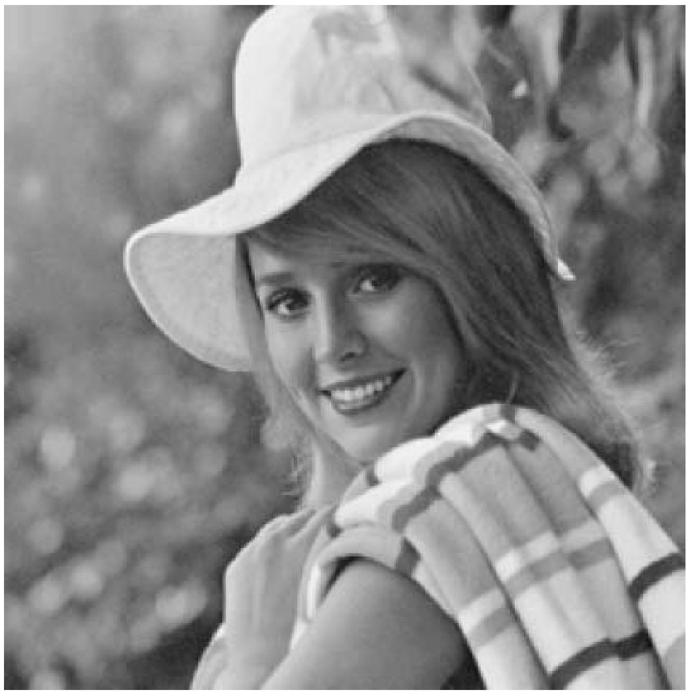

Figure 4.3 Reference image girl.

4.4 Experiments with DCT Features

The main purpose of these experiments was obtaining optimized neural networks. The first issue to train a RBF NN is finding the optimal spread values of the transfer function for the neurons of all RBFNNs. These spread values were found empirically as shown in Figure 4.4. The way of finding the optimal spread value is spanning a range of spread values at which registration errors for test data are computed. In the decision of the optimal spread value, Figure 4.4 is used and the value associated to the minimum of the registration errors in this figure is chosen. Figure 4.4 shows that the search for the optimal spread value of a RBFNN that yields 2000, using the training and test data of reference image ‘girl’. This search was done for 5 dB signal to noise ratio (SNR).

In this manner, an optimal spread value was found for the data generated from each reference image and this computation was done for each SNR value used in the experiments. It can be seen from the plot that registration errors tend to increase slowly after an abrupt fall. If any suboptimal spread value is chosen in a range, it will not increase estimation errors drastically.

All the experiments were done for a noise free image and noisy images with two different SNR values. These SNR values are chosen 20 dB and 5dB for all experiments. For each affine transform parameter, mean of the absolute value of LRFR errors resulted by the RBFNN-based scheme was computed for each affine registration parameter.

The fact that registration errors in scaling factor are very small in value compared to the others, as shown in Table 4.2, should not be mistaken. Since scaling factors in affine transformations are generally small and around 1, in order the estimation of scaling factor to have comparable accuracies, its errors are expected to be much smaller than registration errors in rotation and translations.

In order to make comparisons of performance of RBF NN, mean errors from a FNN with a 20 neurons in one hidden layer by using the same training and test data were also computed and given in the Table 4.2. In the training of FNN, tangent-sigmoid transfer function for the hidden layer neurons and a linear function for the output layer neurons were used. The network was trained using the Levenberg-Marquardt method.

In the training of FNN, the output parameter values fed to the FNN were normalized. This normalization process is done to speed up the convergence in the training stage and this also helped to obtain smaller network output errors in the registration stage. After this process the actual parameter values were obtained by denormalizing the network output values. For a better generalization of the FNN for the test data, we deliberately stopped training early during every training stage. Because of this, the training stage had to be repeated many times with early stopping. This early stopping method continues until it is concluded that a network giving the smallest registration errors for the test data among the trial networks was well enough generalized. The results shown in table 4.2 were obtained after about 25 training trials with early stopping. If the results of a single trial were used, we had had much worse registration errors.

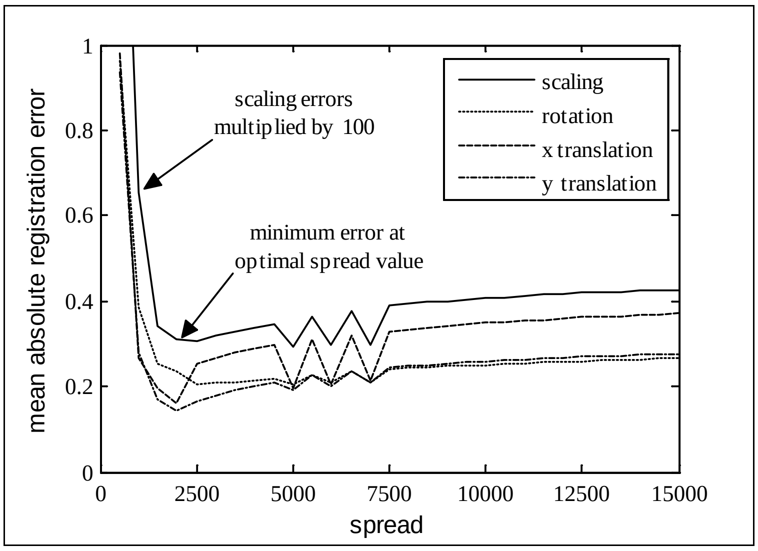

Figure 4.4 Optimal spread value is found by spanning a range of values at which registration errors for test data are computed. Scaling errors were multiplied by 100 to make them visible.

The same procedures explained above for LRFR were repeated for MRCR. The difference of MRCR from LRFR was the dimensions of train and test sets. These sets are also given in table 4.1. The parameter value steps which were used in training had to be large, because it is wanted to keep the size of RBFNN in a moderate level. Mean of the absolute value of MRCR errors resulted by both networks are given in Table 4.3. From the table 4.3, it can be said that the mean errors are relatively high. However, the estimated registration parameters are in an acceptable range for a Coarse registration task.

It must be taken into consideration that, such a Coarse registration task is to be followed by a fine registration performed using a neural network trained in LRFR mode. The LRFR process was done after back transforming the image of the Coarse registration.

In the MRCR mode, coarse registration parameters that are estimated must be in the range for which that fine registration network is capable of estimating fine registration parameters accurately. This can be done by selecting a proper range and also sufficiently small step size for registration parameter values used in the training stage of that coarse registration network.

The results of these experiments are given in table 4.2 and table 4.3. These results show that RBFNN-based registration scheme is more accurate in estimating the affine transformation parameters for both LRFR and MRCR. For LRFR, it can be said that the accuracies of both of the schemes becomes nearly the same as SNR reduces to around 20 dB. However, for MRCR, the accuracy of the RBFNN-based registration scheme is clearly much better compared to FNN-based scheme. This is because of the larger parameter value steps in the MRCR training set. For this reason, network approximation errors are inevitably become large and dominate noise errors at the output of any of the two networks. Because of too small parameter value steps in the training set, LRFR estimates the parameter values very accurate. In this mode, still RBFNN-based scheme outperforms the FNN-based scheme at the high SNR values. Inherently, as the noise strength increases noise errors start dominating network approximation errors. For this reason in relatively low SNR values, there exist similar estimation accuracies for both schemes because an estimator in the highest accuracy region is prone to larger degradations caused by noise. In LRFR, the FNN-based scheme seems to be more robust to noise but this is not a valid fact for the total registration accuracy. Table 4.3 also shows that the RBFNN-based scheme is more robust to additive Gaussian noise in MRCR in addition to being more accurate.

Table 4.2 Mean absolute registration errors of RBFNN-based and FNN-based schemes for LRFR for DCT features

Image	SNR	Neural Network	scaling	rotation	horizontal translation	vertical translation
aerial	Noise free	RBF NN	0.0002	0.01	0.007	0.02
		FNN	0.0004	0.04	0.02	0.03
	20 dB	RBF NN	0.0006	0.06	0.03	0.05
		FNN	0.001	0.1	0.04	0.07
	5 dB	RBF NN	0.002	0.18	0.1	0.15
		FNN	0.003	0.22	0.17	0.25
moon	Noise free	RBF NN	0.0003	0.03	0.004	0.01
		FNN	0.0006	0.06	0.01	0.02
	20 dB	RBF NN	0.001	0.09	0.06	0.04
		FNN	0.001	0.1	0.05	0.06
	5 dB	RBF NN	0.003	0.2	0.18	0.18
		FNN	0.004	0.3	0.24	0.19
girl	Noise free	RBF NN	0.0001	0.005	0.004	0.004
		FNN	0.0003	0.01	0.01	0.02
	20 dB	RBF NN	0.0007	0.05	0.05	0.03
		FNN	0.0009	0.05	0.04	0.05
	5 dB	RBF NN	0.003	0.2	0.17	0.15
		FNN	0.003	0.2	0.15	0.19

Table 4.3 Mean absolute registration errors of RBFNN-based and FNN-based schemes for MRCR for DCT features

Image	SNR	Neural Network	scaling	rotation	horizontal translation	vertical translation
aerial	Noise free	RBF	0.014	1.3	1.0	1.3
		FNN	0.022	1.7	1.2	1.6
	20 dB	RBF	0.014	1.3	1.0	1.3
		FNN	0.022	1.7	1.1	1.4
	5 dB	RBF	0.018	1.5	1.1	1.5
		FNN	0.032	2.2	1.7	2.2
moon	Noise free	RBF	0.014	1.0	0.5	0.5
		FNN	0.021	1.7	0.9	0.8
	20 dB	RBF	0.017	1.4	0.6	0.6
		FNN	0.025	1.8	1.0	0.9
	5 dB	RBF	0.023	1.3	1.1	0.9
		FNN	0.031	2.4	1.5	1.3
girl	Noise free	RBF	0.011	0.8	0.9	0.7
		FNN	0.018	1.2	1.2	1.1
	20 dB	RBF	0.011	0.8	0.8	0.9
		FNN	0.020	1.4	1.3	1.0
	5 dB	RBF	0.013	1.0	0.9	0.8
		FNN	0.021	1.6	1.4	1.5

4.5 Experiments with 2D PCA Features

In order to compare the results that were obtained for the DCT features, it was needed to use a different feature extraction method, by the way, different features to train the neural networks. For this reason Two Dimensional Principal Component Analysis (2D-PCA) was used as the feature extraction transform.

Again, the same procedure was applied to the RBFNN and FNN. The optimal spread values of the transfer function for the neurons of all RBFNNs were found empirically. After that, the training and testing processes were completed and mean of the absolute value of LRFR errors resulted by the RBFNN-based scheme was computed for each affine registration parameter, and the computed error values were given in Table 4.4 for the same SNR values as DCT features. Also, the training and testing processes were completed for the FNN same as it was done in DCT features. Mean errors computed from a FNN with a 20 neurons in one hidden layer by using the same training and test data were also given in the Table 4.4.

While training in the FNN, In order to improve generalization of the FNN for the test data during every training stage, training was deliberately stopped early. For this reason, training stage had to be repeated with early stopping as many times until having the smallest registration errors for the test data. The registration error results for the FNN-based scheme given in Table 4.4 were obtained after about 25 training trials with early stopping. In a single trial, an average FNN gives much worse registration errors.

The same procedures explained for LRFR of were repeated for MRCR. The only difference of these two modes is the difference of the transformation parameter values used for the training and test sets. All the values are shown in Table 4.1. As it was in the DCT parameters, the size of the RBFNN is wanted to be kept moderate. For this reason, the parameter value steps chosen for training had to be large at the cost of much lower accuracy in estimating the registration parameters. Mean of the absolute value of MRCR errors resulted by both networks are given in Table 4.5. In fact, the mean errors for each parameter are relatively high, but if we take into consideration that this is a coarse registration task, then, the registration parameters are in an acceptable range.

According to the experimental results in table 4.4 and table 4.5, we can say that RBFNN is much better than FNN for the 2DPCA features. However if we compare DCT versus 2DPCA features, DCT has a better performance in noise free condition and higher SNR values.

Table 4.4 Mean absolute registration errors of RBFNN-based and FNN-based schemes for LRFR for 2DPCA features

Image	SNR	Neural Network	scaling	rotation	horizontal translation	vertical translation
aerial	Noise free	RBF NN	0.0008	0.0694	0.0522	0.0740
		FNN	0.0023	0.1472	0.1036	0.1625
	20 dB	RBF NN	0.0008	0.0694	0.0694	0.0843
		FNN	0.0018	0.1551	0.1177	0.1829
	5 dB	RBF NN	0.0018	0.1302	0.1156	0.1461
		FNN	0.0040	0.2289	0.2106	0.2541
moon	Noise free	RBF NN	0.0020	0.0882	0.1249	0.0622
		FNN	0.0023	0.1240	0.0943	0.0681
	20 dB	RBF NN	0.0013	0.0831	0.0451	0.0655
		FNN	0.0018	0.1249	0.1315	0.0845
	5 dB	RBF NN	0.0025	0.1487	0.1276	0.1002
		FNN	0.0041	0.2595	0.2593	0.2195
girl	Noise free	RBF NN	0.0006	0.0767	0.0395	0.0279
		FNN	0.0015	0.0822	0.0653	0.0654
	20 dB	RBF NN	0.0008	0.0638	0.0508	0.0492
		FNN	0.0009	0.1132	0.0717	0.0723
	5 dB	RBF NN	0.0020	0.1774	0.1544	0.1135
		FNN	0.0034	0.2037	0.1779	0.1936

Table 4.5 Mean absolute registration errors of RBFNN-based and FNN-based schemes for MRCR for 2DPCA features

Image	SNR	Neural Network	scaling	rotation	horizontal translation	vertical translation
aerial	Noise free	RBF NN	0.021	1.480	1.186	1.129
		FNN	0.028	3.430	2.143	2.872
	20 dB	RBF NN	0.021	1.505	1.149	1.112
		FNN	0.035	3.361	1.808	2.393
	5 dB	RBF NN	0.023	1.476	1.145	1.147
		FNN	0.033	3.440	1.920	2.470
moon	Noise free	RBF NN	0.012	1.40	0.67	0.59
		FNN	0.012	1.859	1.062	1.094
	20 dB	RBF NN	0.013	1.390	0.676	0.795
		FNN	0.019	1.752	1.163	1.013
	5 dB	RBF NN	0.014	1.449	1.023	0.946
		FNN	0.020	2.109	1.283	1.086
girl	Noise free	RBF NN	0.010	0.677	0.619	0.472
		FNN	0.022	1.645	0.990	1.190
	20 dB	RBF NN	0.011	0.672	0.639	0.481
		FNN	0.020	1.610	1.077	1.010
	5 dB	RBF NN	0.012	0.768	0.784	0.604
		FNN	0.021	1.382	1.099	1.012

4.6 Performance Comparisons

As mentioned in the former parts, in this study, experiments done in MATLAB by using two kinds of feature extraction methods and two kinds of neural networks; that is DCT and 2DPCA were used as feature extraction methods and RBF NN and FNN were used as artificial neural network structures. In this part, the performances of two networks used and two transforms used are compared respectively. While comparing these results, graphs based on table 4.2, table 4.3 table 4.4 and table 4.5 were used in order to understand the results better.

4.6.1 Characteristics of the Computer Used

In this thesis, the computer used for the experiments done has a Intel Core 2 Quad 2.5 GHz CPU. The random access memory (RAM) of the computer has a capacity of 4 GB. The computer uses Windows 7 Ultimate as the operating system. In the experiments, MATLAB R2009a was used as the version of MATLAB.

4.6.2 RBF NN and FNN Performance Comparison

The experiments done in all the study have two main categories. These are local range fine registration (LRFR) and medium range coarse registration (MRCR). The graphs in the Figures from 4.5 to 4.22 refer to LRFR category and Figures from 4.23 to 4.40 refer to MRCR category. In these graphs, the performance of RBF NN and FNN are compared both for DCT and 2DPCA features. In the comparisons, it was seen that in almost all the varieties of noise conditions that is, noise free, 20 db noise and 5 db noise, RBF NN has a superior performance compared to FNN independent of the image used. As SNR is reduces to around 20 db, LRFR accuracies of both registration schemes become nearly the same. On the other hand MRCR accuracy of the RBFNN-based registration scheme is much better than the FNN-based registration scheme. The graphs from 4.5 to 4.40 are given in the next parts 4.6.2.1., 4.6.2.2., 4.6.2.3 and 4.6.2.4. At the end of this part, in Table 4.6 the comparison of RBF NN and FNN is also given as a summary of this section.

4.6.2.1 The Results of LRFR of DCT

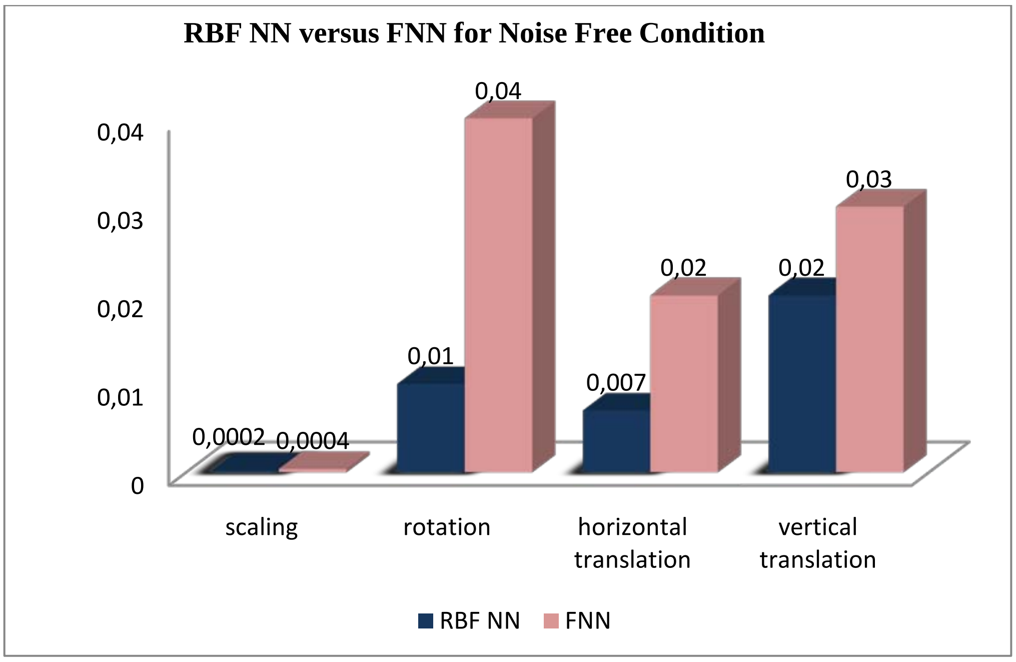

Figure 4.5 The errors of RBF NN versus FNN for noise free condition for the image aerial for DCT features in LRFR

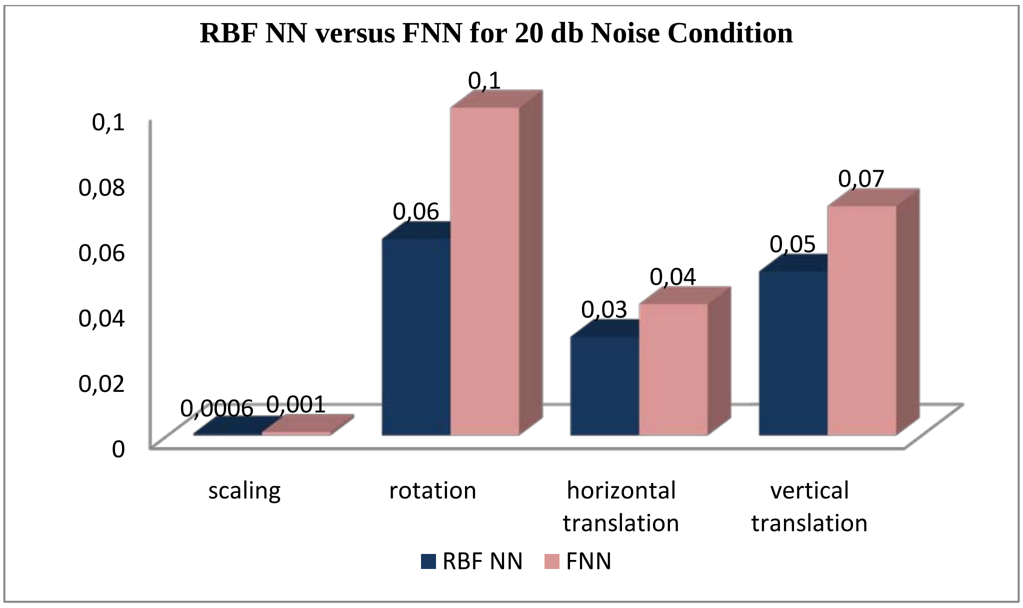

Figure 4.6 The errors of RBF NN versus FNN for 20 db Noise Condition for the image aerial and DCT features in LRFR

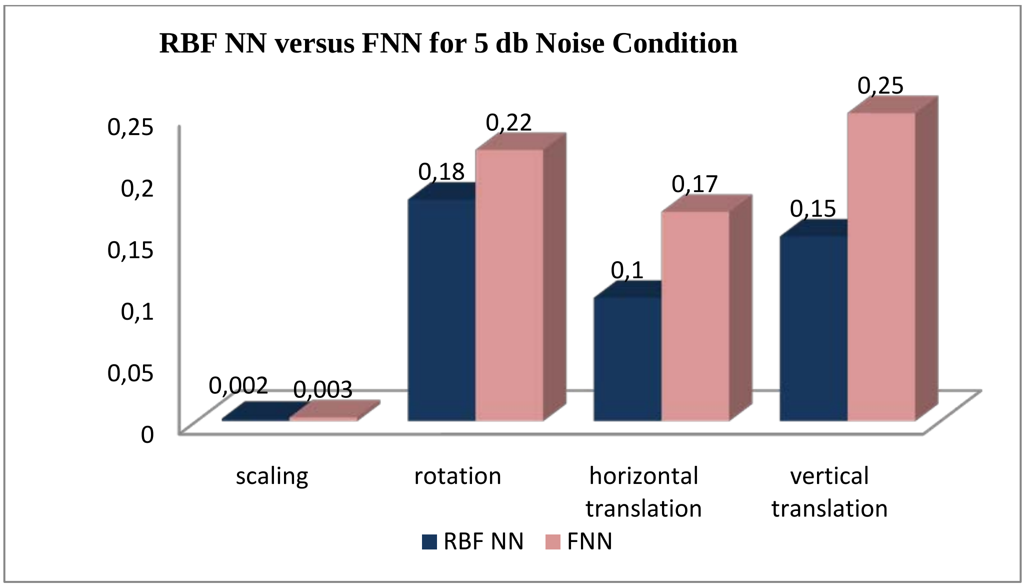

Figure 4.7 The errors of RBF NN versus FNN for 5 db Noise Condition for the image aerial and DCT features in LRFR

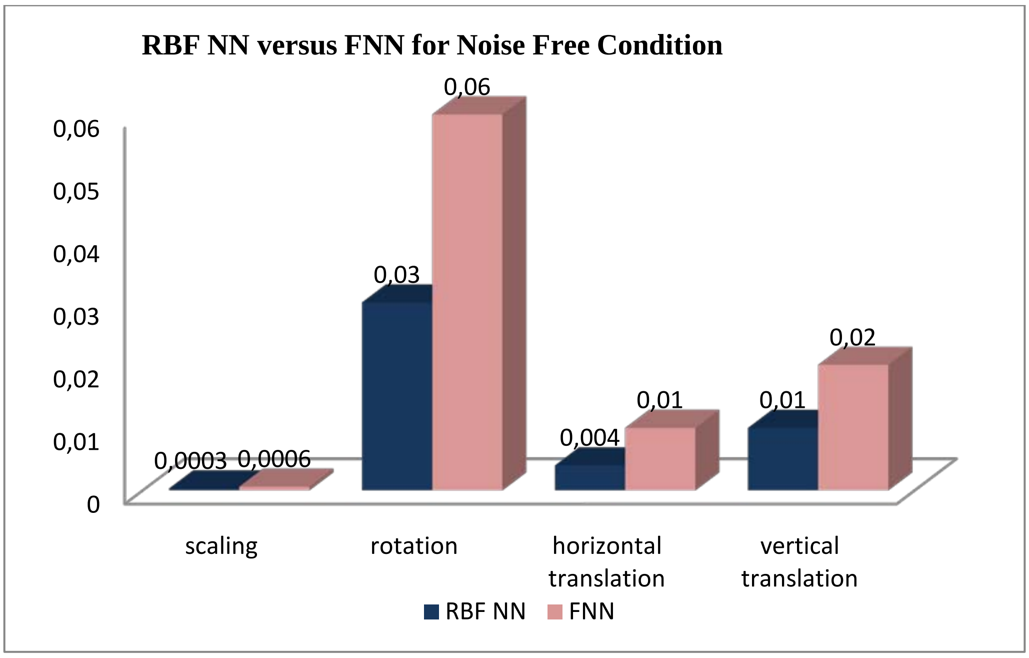

Figure 4.8 The errors of RBF NN versus FNN for Noise Free Condition for the image moon and DCT features in LRFR

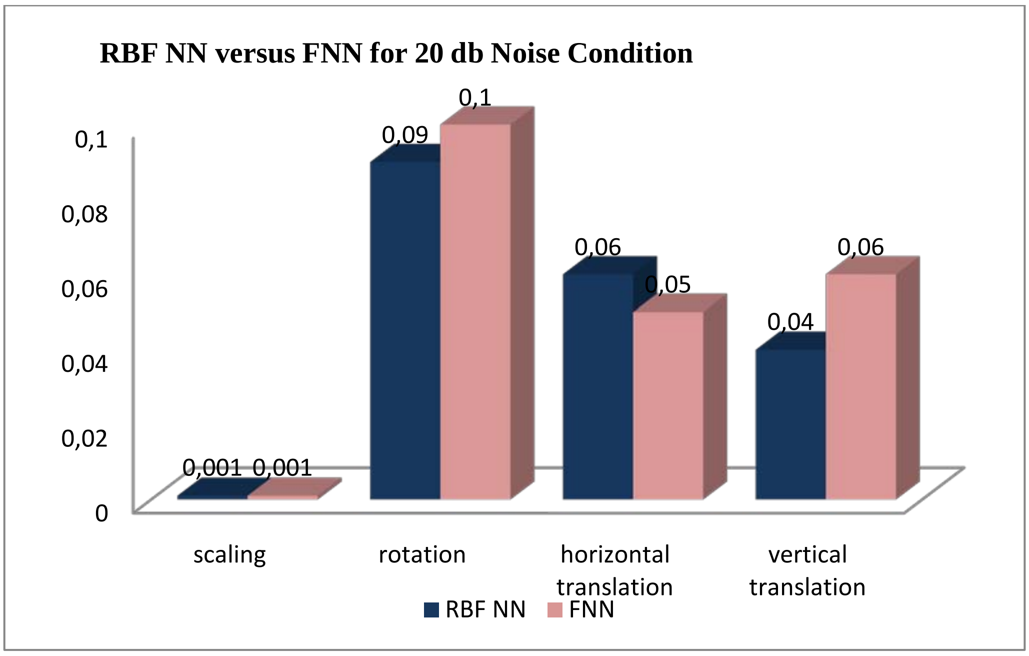

Figure 4.9 The errors of RBF NN versus FNN for 20 db Noise Condition for the image moon and DCT features in LRFR

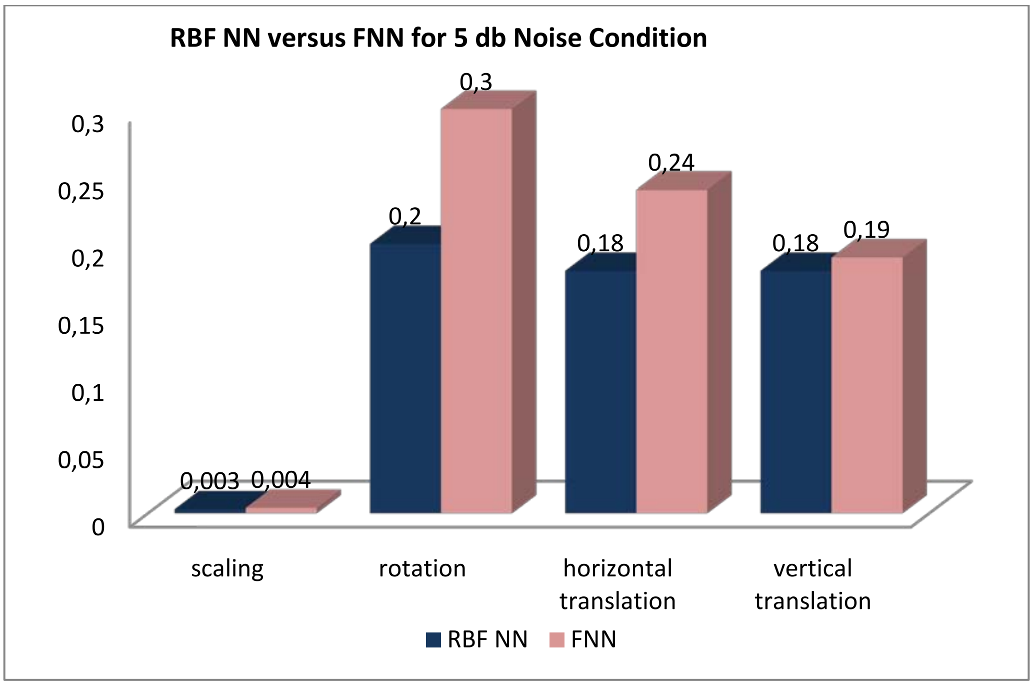

Figure 4.10 The errors of RBF NN versus FNN for 5 db Noise Condition for the image moon and DCT features in LRFR

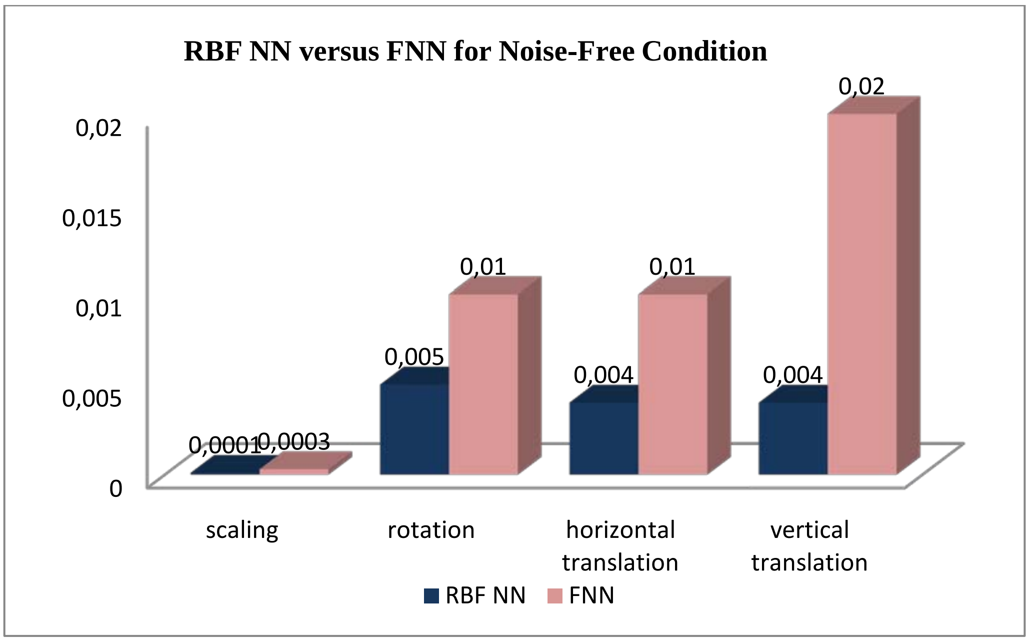

Figure 4.11 The errors of RBF NN versus FNN for Noise-Free Condition for the image girl and DCT features in LRFR

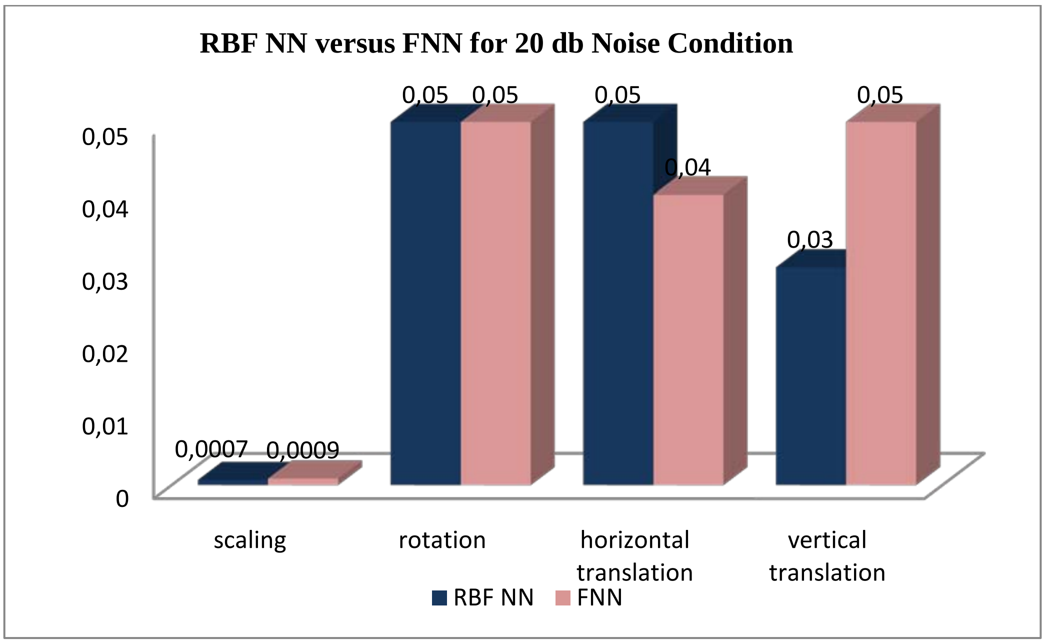

Figure 4.12 The errors of RBF NN versus FNN for 20 db noise Condition for the image girl and DCT features in LRFR

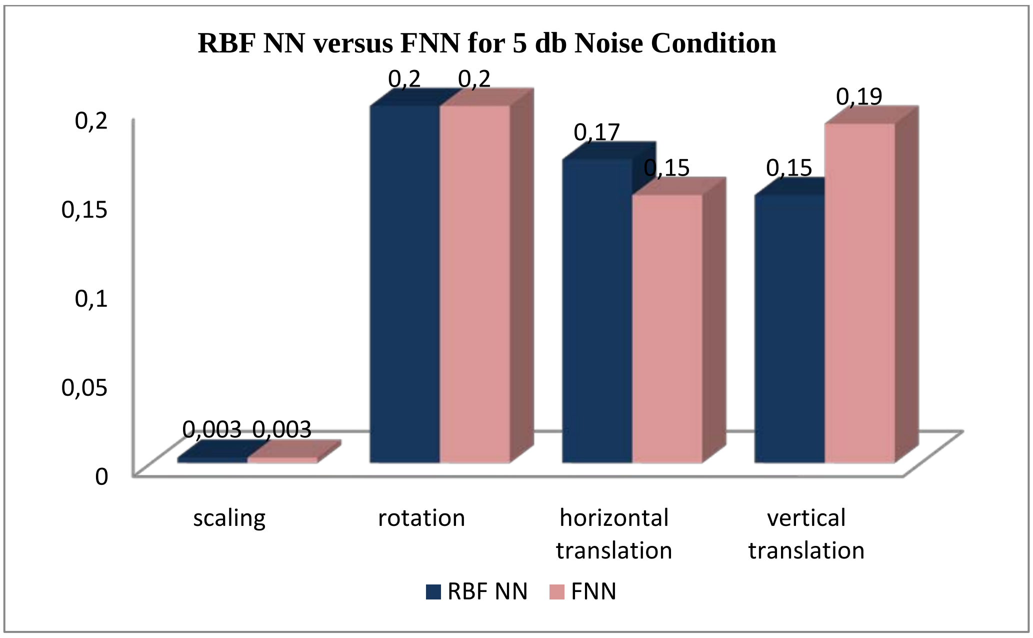

Figure 4.13 The errors of RBF NN versus FNN for 5 db noise Condition for the image girl and DCT features in LRFR

4.6.2.2 The Results of LRFR of 2DPCA

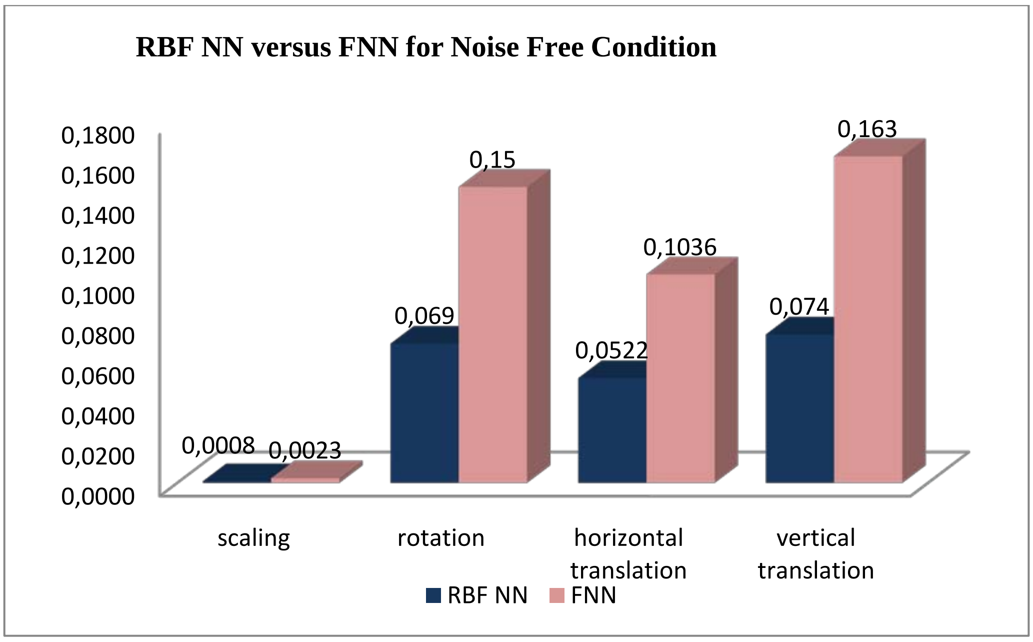

Figure 4.14 The errors of RBF NN versus FNN for noise free condition for the image aerial for 2DPCA features in LRFR

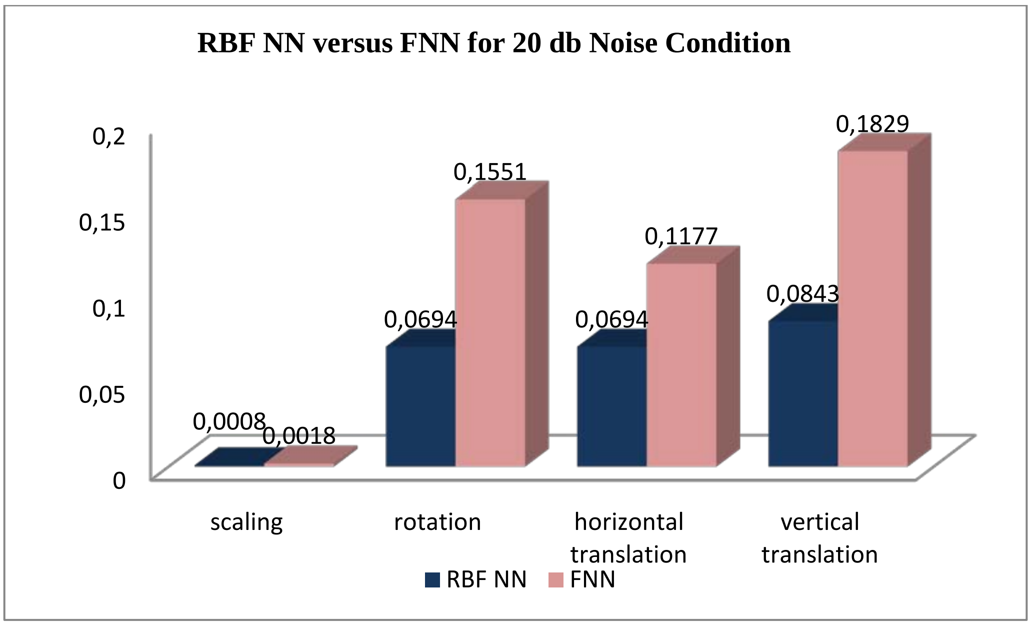

Figure 4.15 The errors of RBF NN versus FNN for 20 db noise condition for the image aerial for 2DPCA features in LRFR

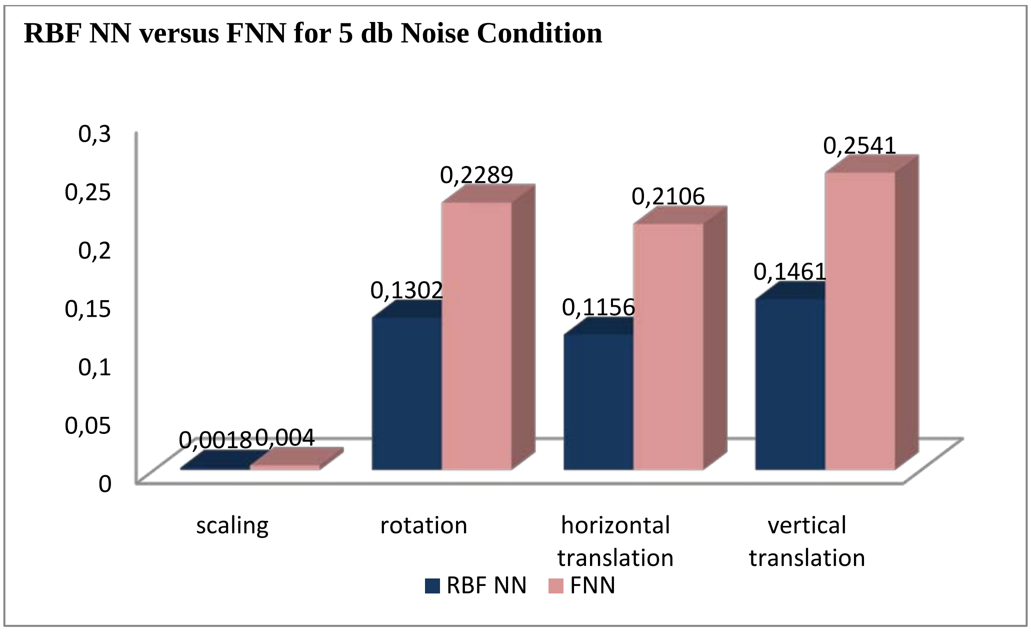

Figure 4.16 The errors of RBF NN versus FNN for 5 db noise condition for the image aerial for 2DPCA features in LRFR

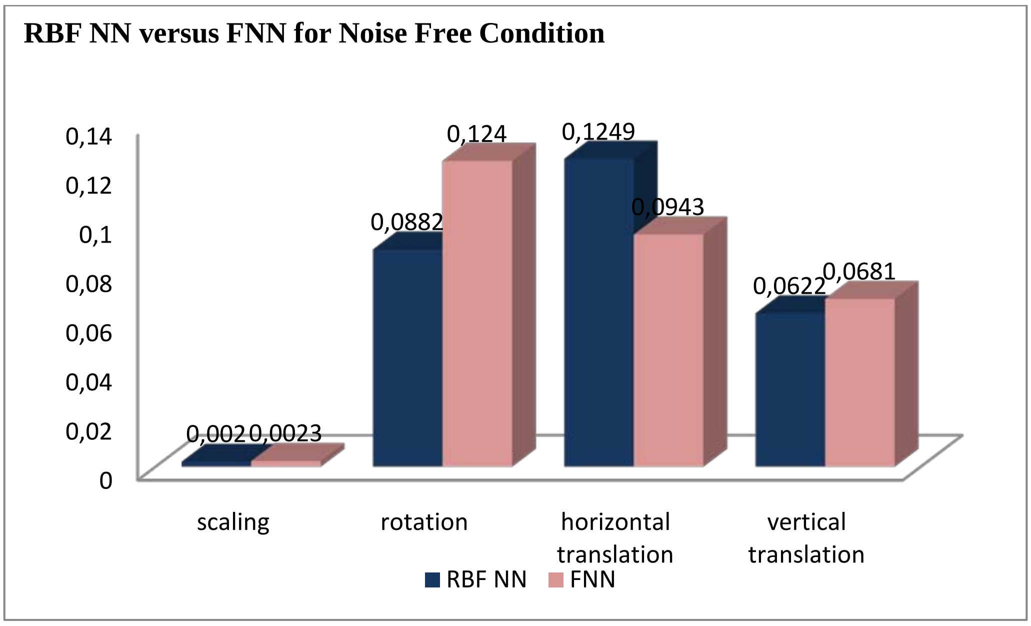

Figure 4.17 The errors of RBF NN versus FNN for noise free condition for the image moon for 2DPCA features in LRFR

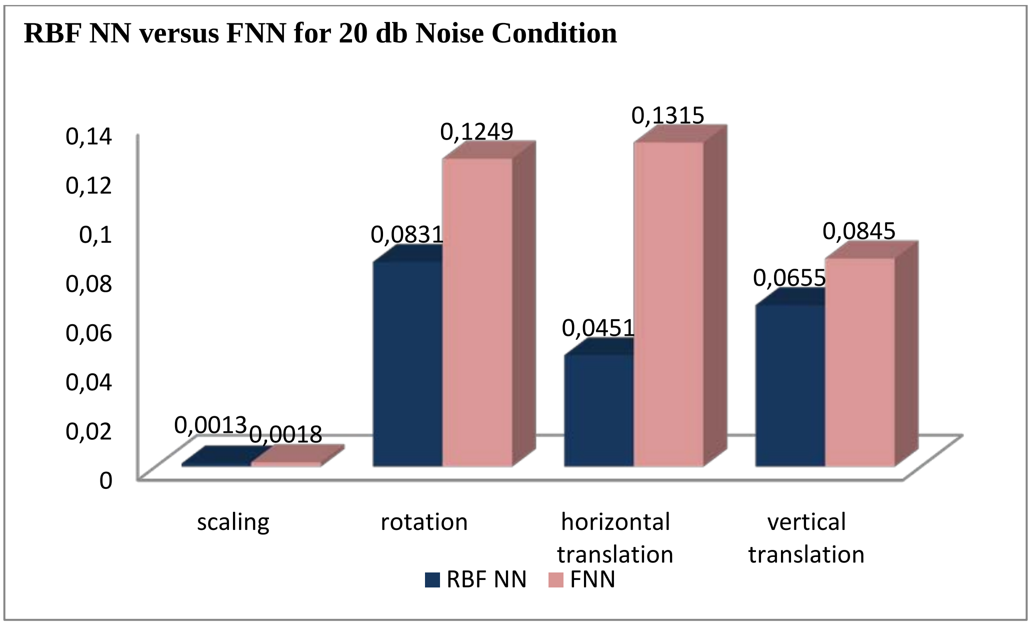

Figure 4.18 The errors of RBF NN versus FNN for 20 db noise condition for the image moon for 2DPCA features in LRFR

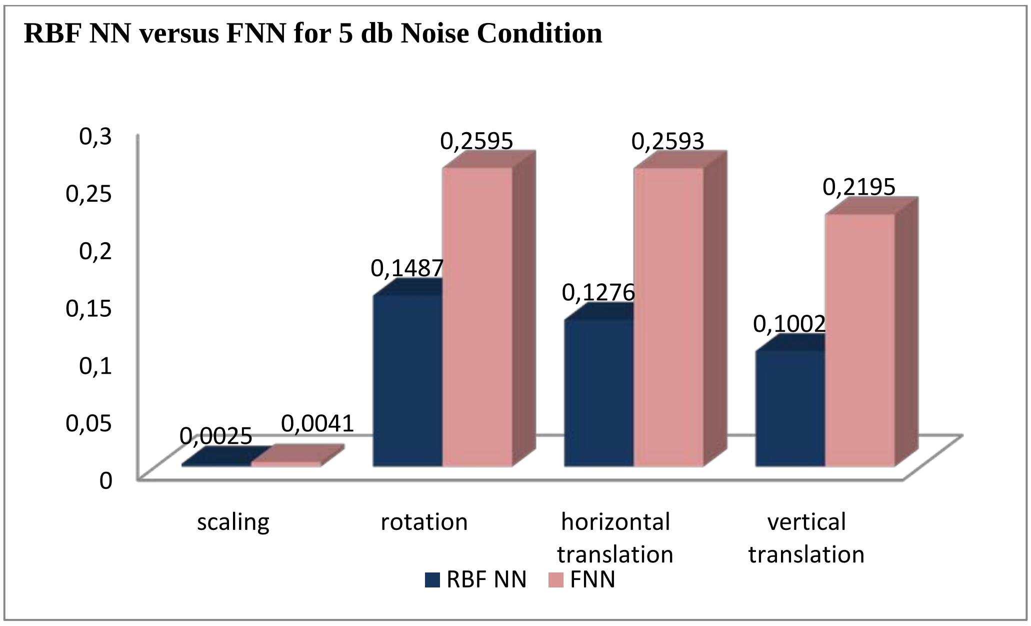

Figure 4.19 The errors of RBF NN versus FNN for 5 db noise condition for the image moon for 2DPCA features in LRFR

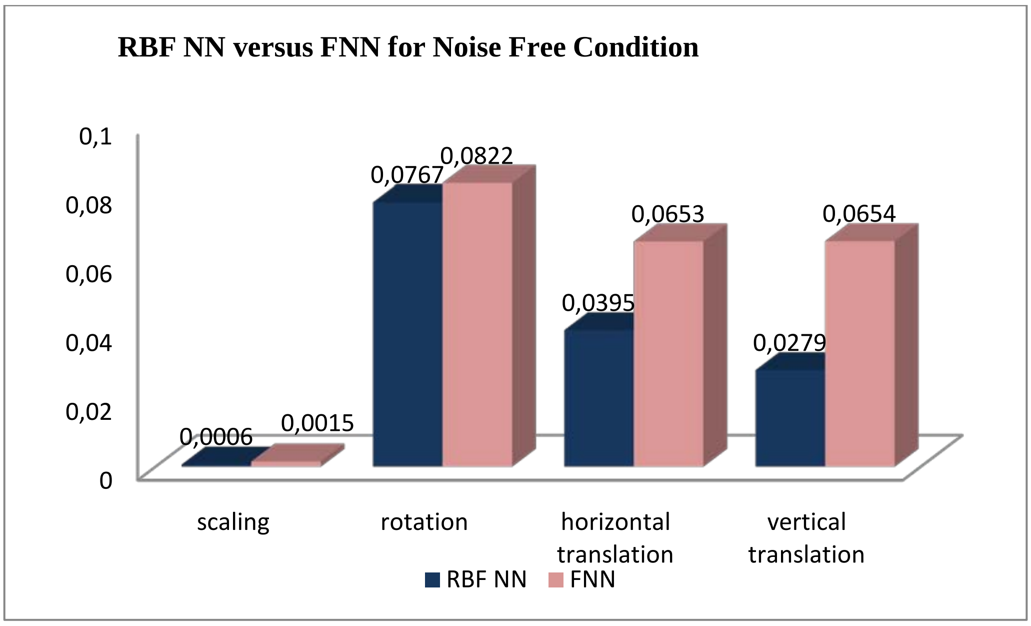

Figure 4.20 The errors of RBF NN versus FNN for noise free condition for the image girl for 2DPCA features in LRFR

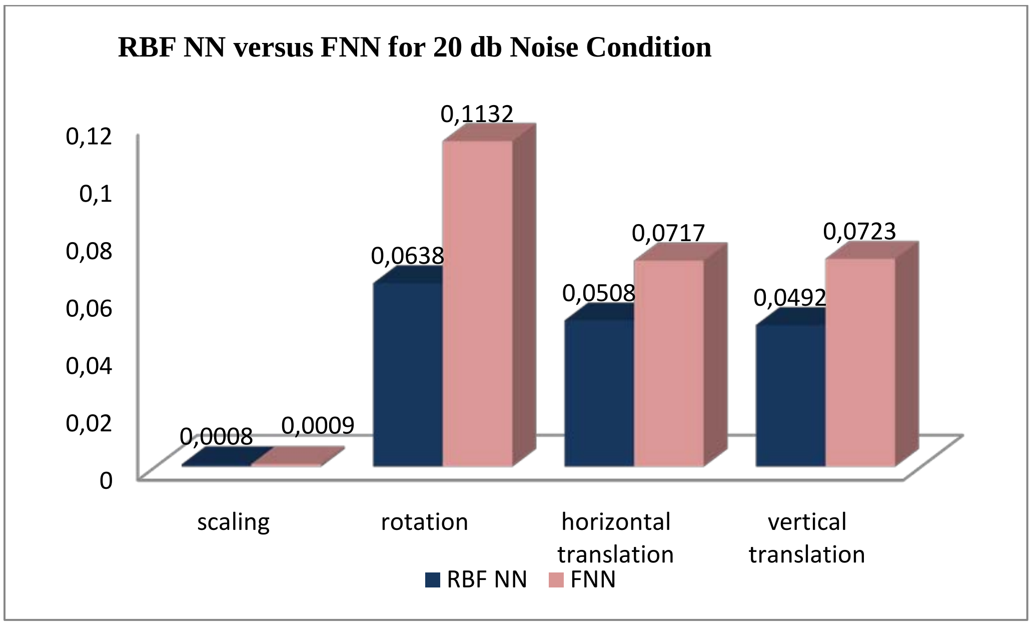

Figure 4.21 The errors of RBF NN versus FNN for 20 db noise condition for the image girl for 2DPCA features in LRFR

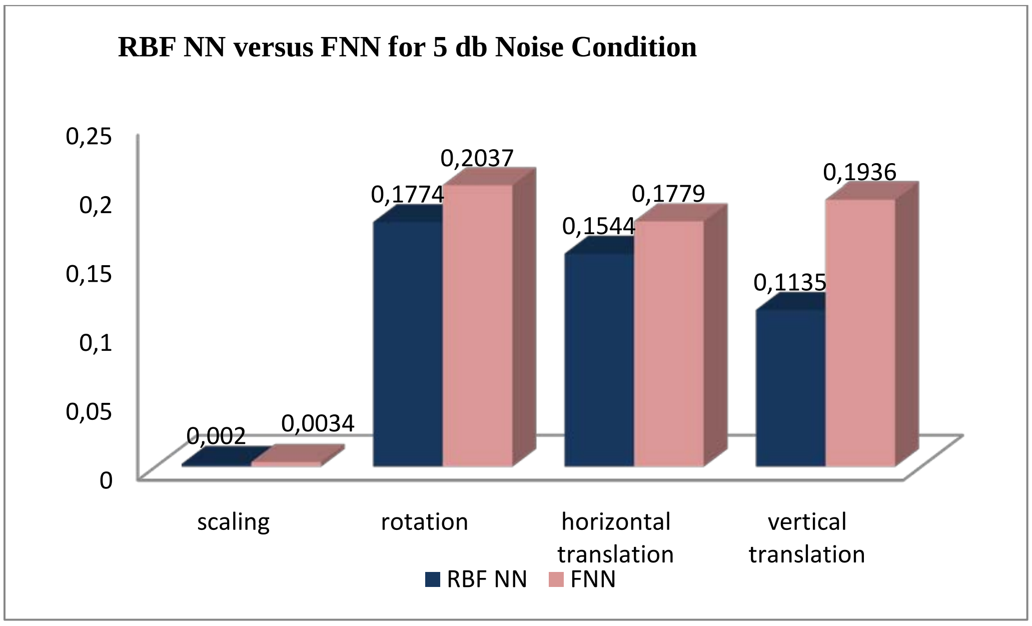

Figure 4.22 The errors of RBF NN versus FNN for 5 db noise condition for the image girl for 2DPCA features in LRFR

4.6.2.3 The Results of MRCR of DCT

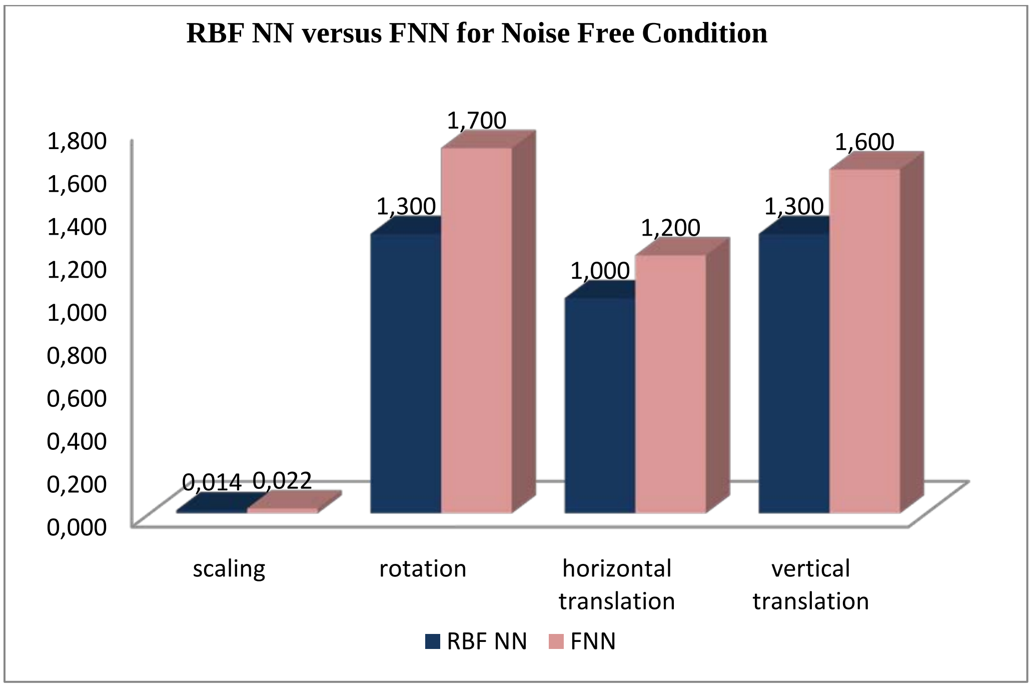

Figure 4.23 The errors of RBF NN versus FNN for noise free condition for the image aerial for DCT features in MRCR

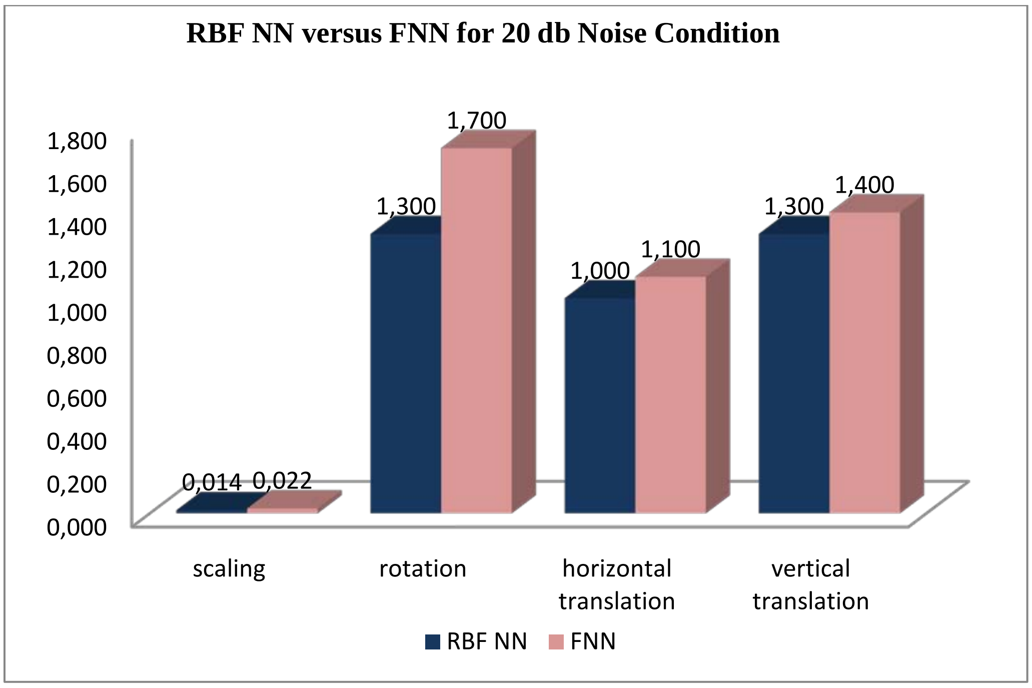

Figure 4.24 The errors of RBF NN versus FNN for 20 db noise condition for the image aerial for DCT features in MRCR

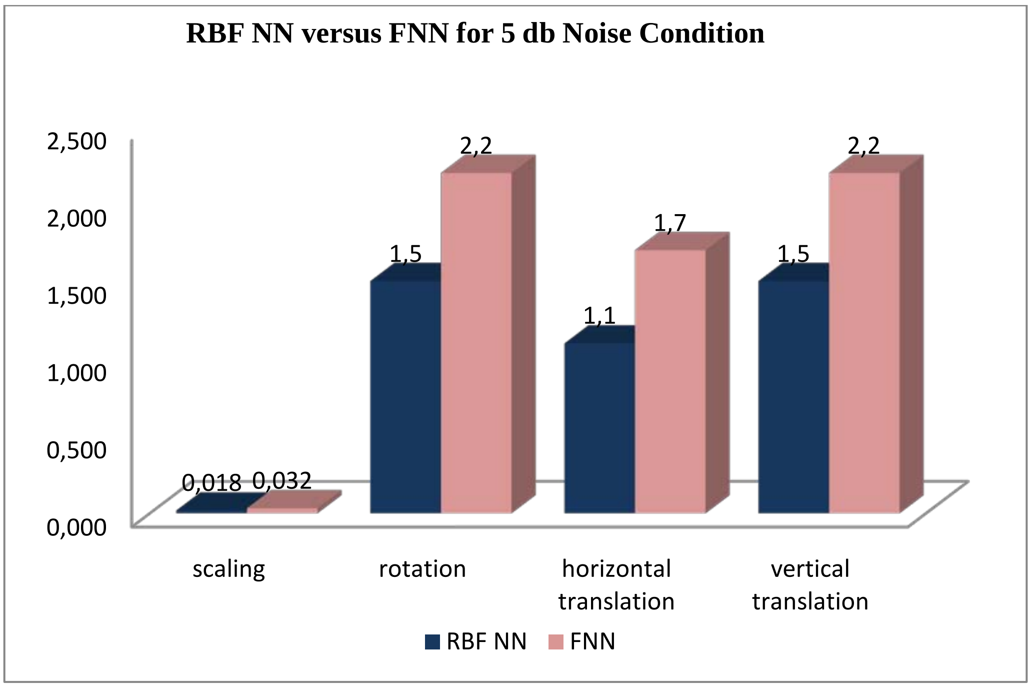

Figure 4.25 The errors of RBF NN versus FNN for 5 db noise condition for the image aerial for DCT features in MRCR

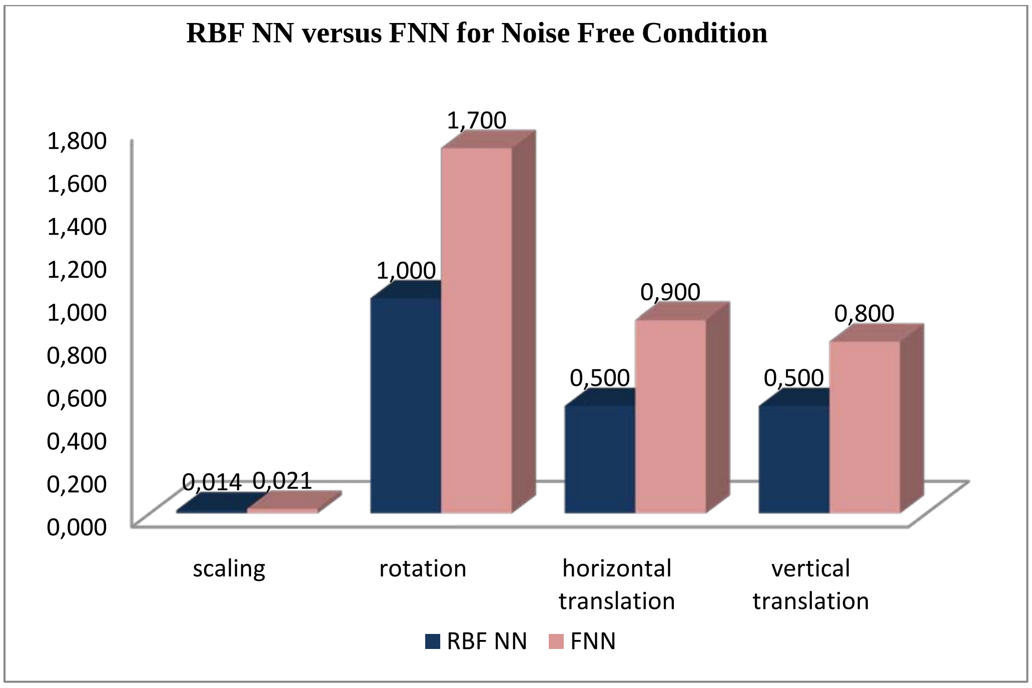

Figure 4.26 The errors of RBF NN versus FNN for noise free condition for the image moon for DCT features in MRCR

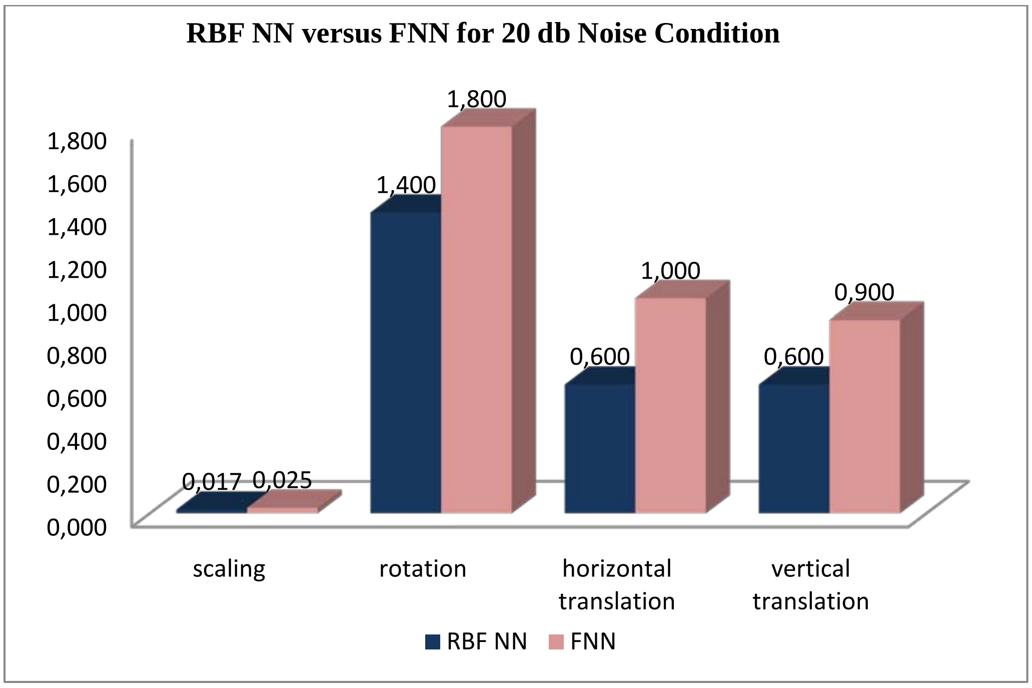

Figure 4.27 The errors of RBF NN versus FNN for 20 db noise condition for the image moon for DCT features in MRCR

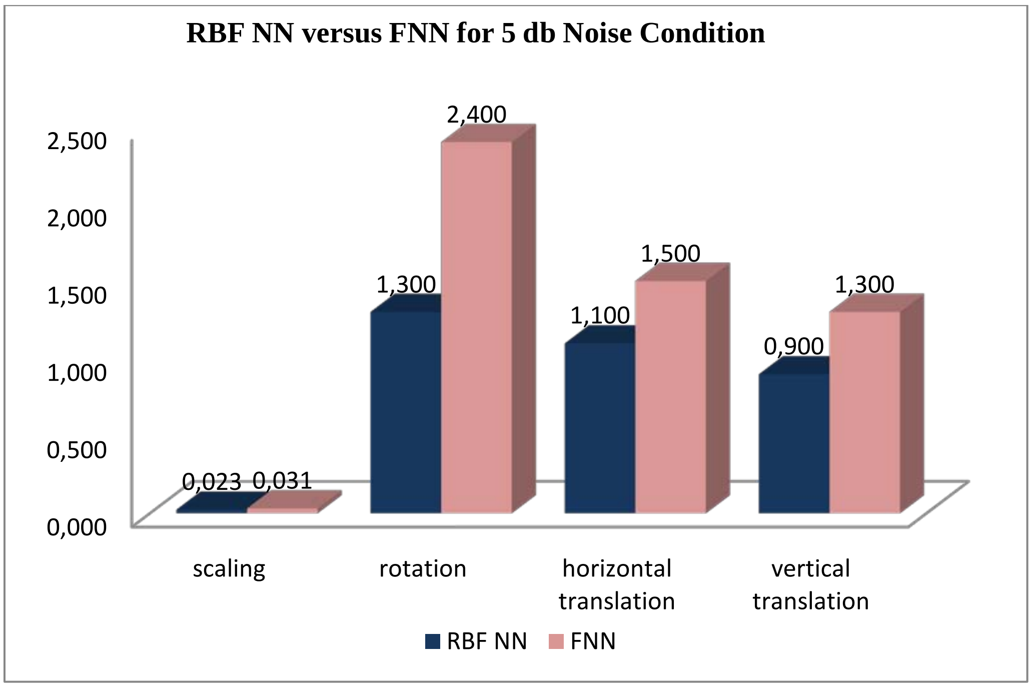

Figure 4.28 The errors of RBF NN versus FNN for 5 db noise condition for the image moon for DCT features in MRCR

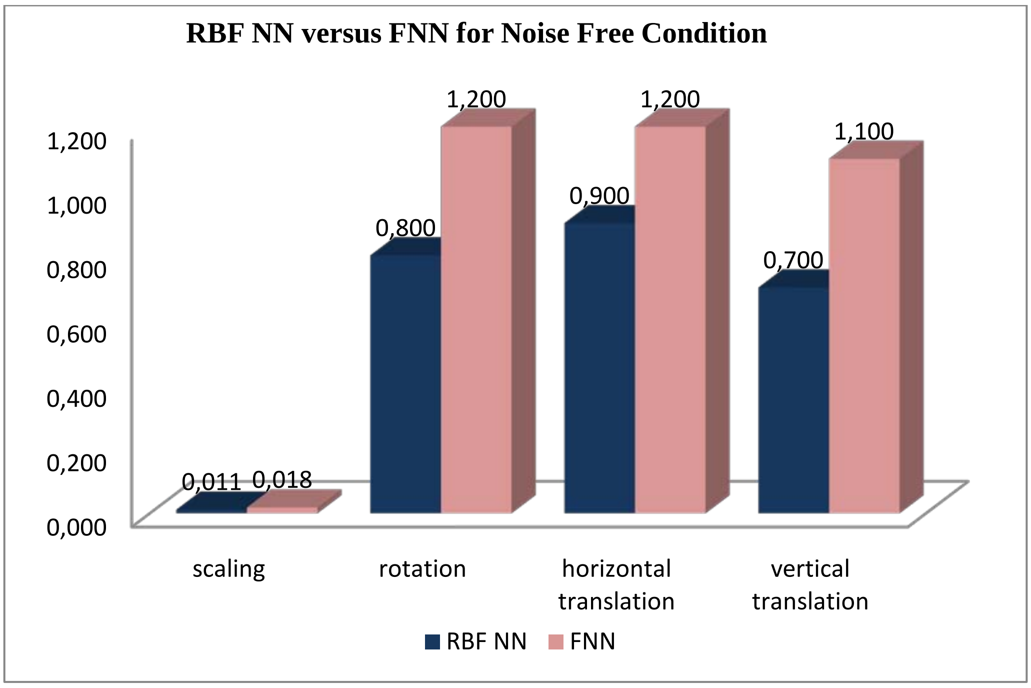

Figure 4.29 The errors of RBF NN versus FNN for noise free condition for the image girl for DCT features in MRCR

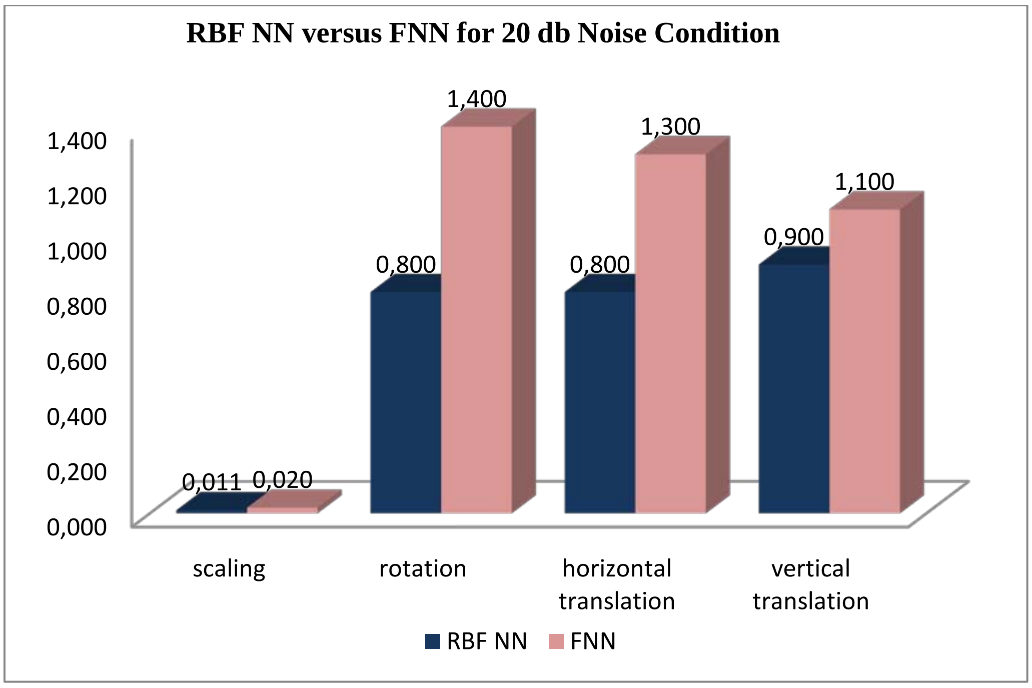

Figure 4.30 The errors of RBF NN versus FNN for 20 db noise condition for the image girl for DCT features in MRCR

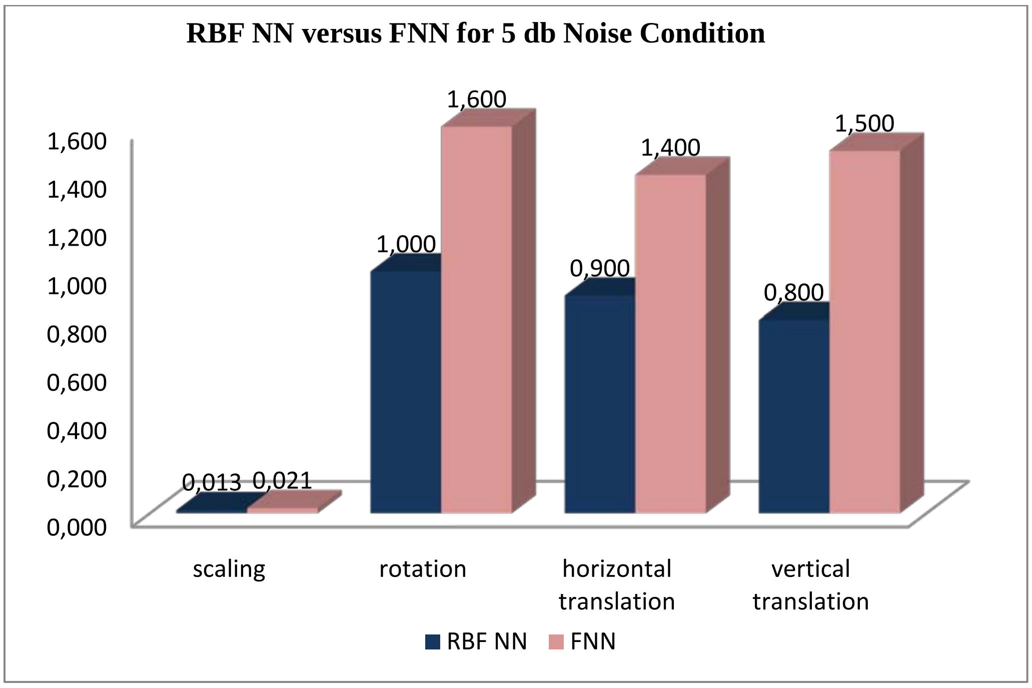

Figure 4.31 The errors of RBF NN versus FNN for 5 db noise condition for the image girl for DCT features in MRCR

4.6.2.4 The Results of MRCR of 2DPCA

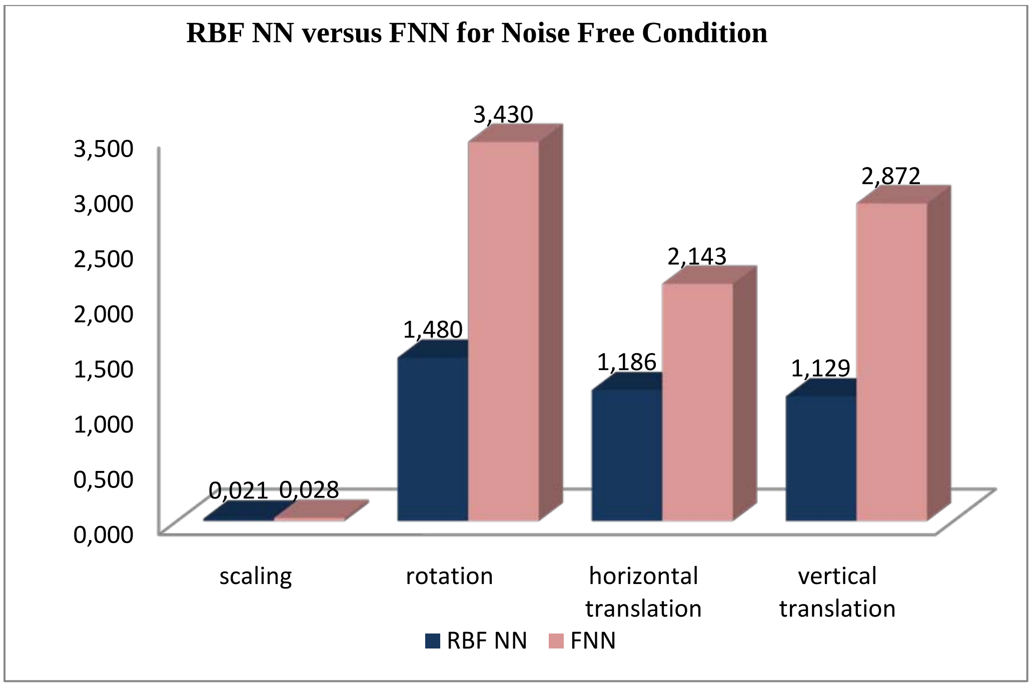

Figure 4.32 The errors of RBF NN versus FNN for noise free condition for the image aerial for 2DPCA features in MRCR

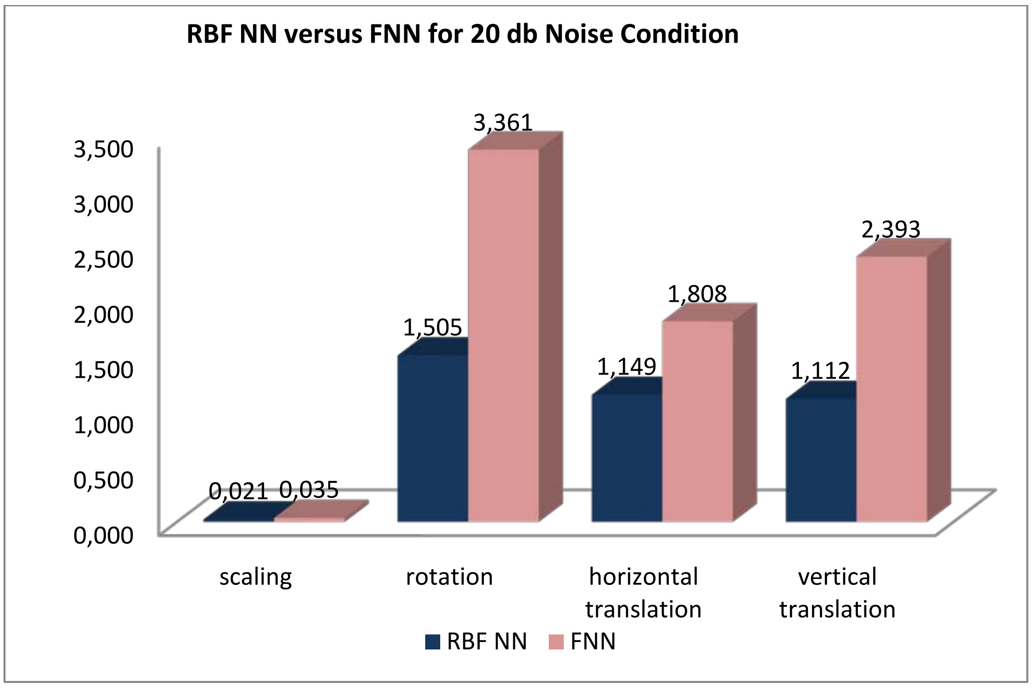

Figure 4.33 The errors of RBF NN versus FNN for 20 db noise condition for the image aerial for 2DPCA features in MRCR

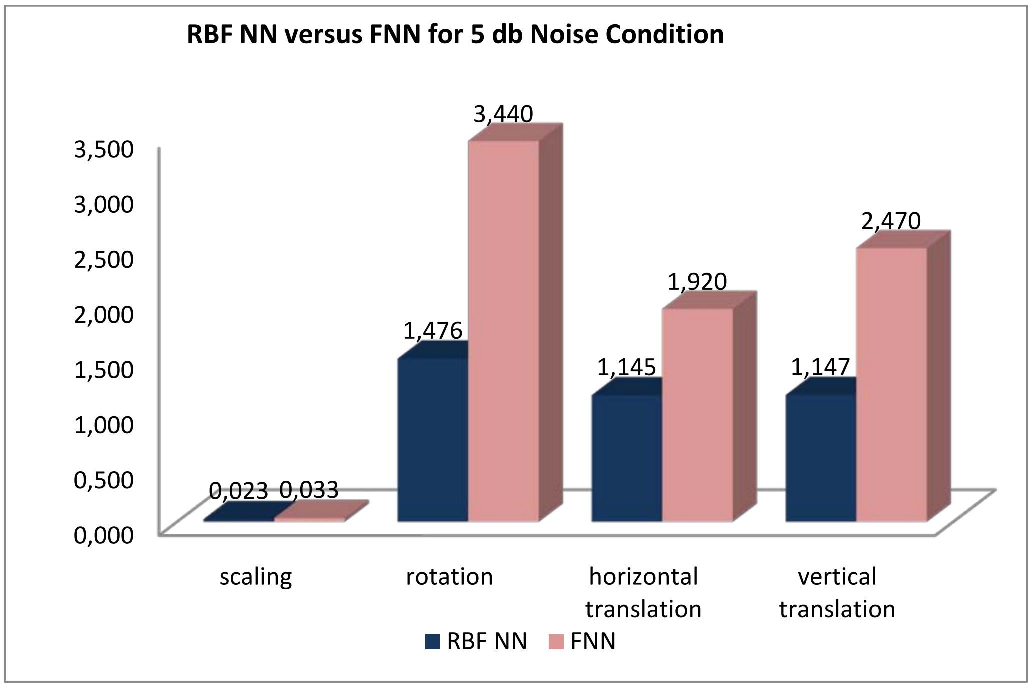

Figure 4.34 The errors of RBF NN versus FNN for 5 db noise condition for the image aerial for 2DPCA features in MRCR

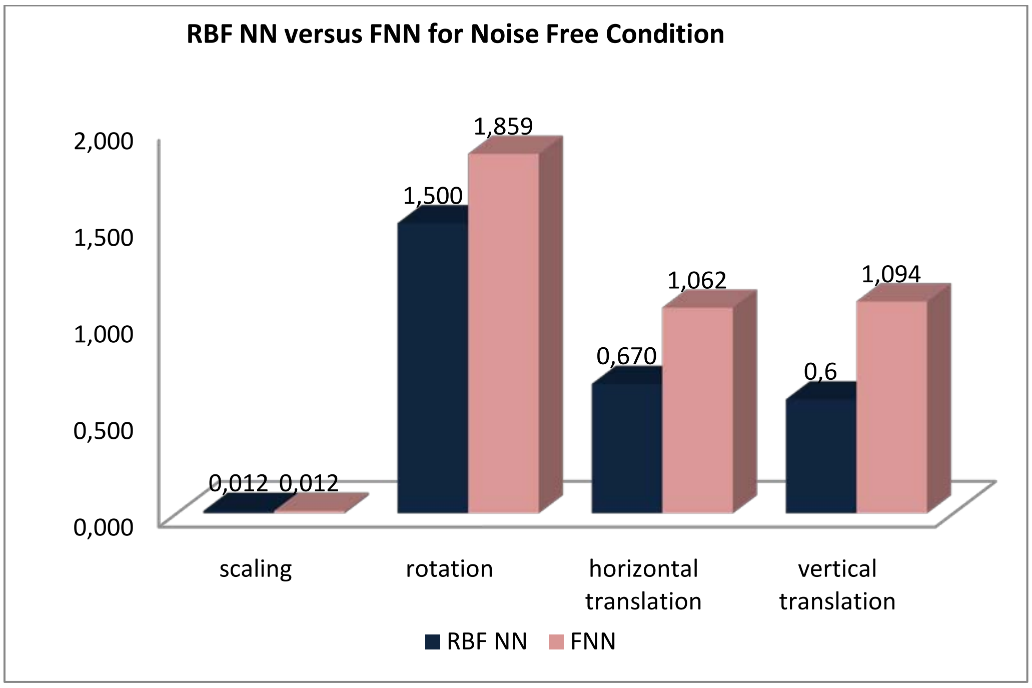

Figure 4.35 The errors of RBF NN versus FNN for noise free condition for the image moon for 2DPCA features in MRCR

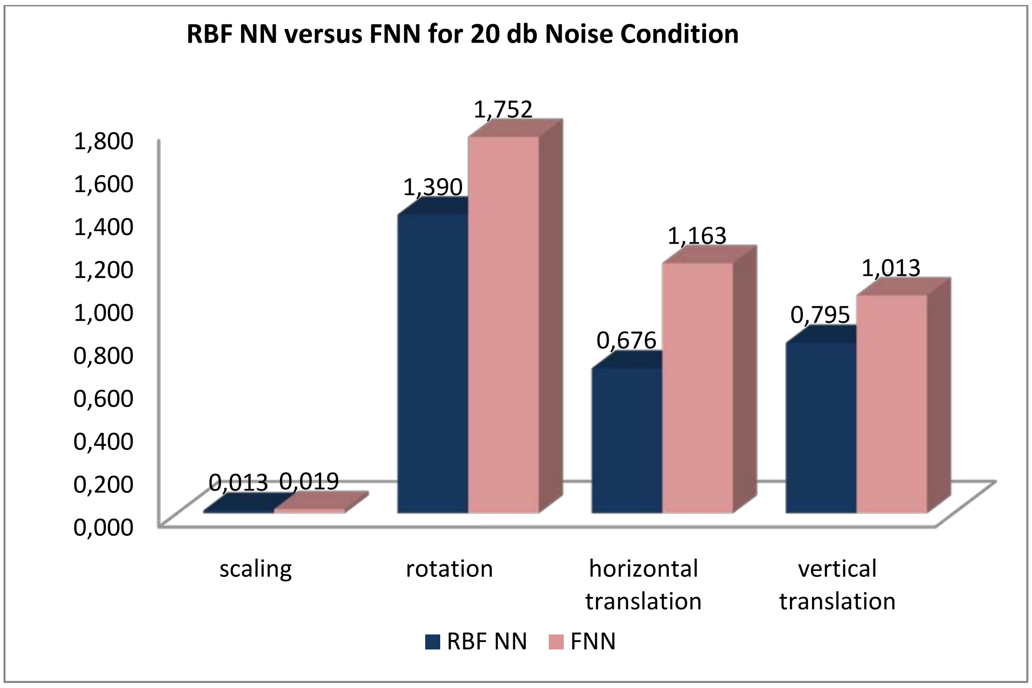

Figure 4.36 The errors of RBF NN versus FNN for 20 db noise condition for the image moon for 2DPCA features in MRCR

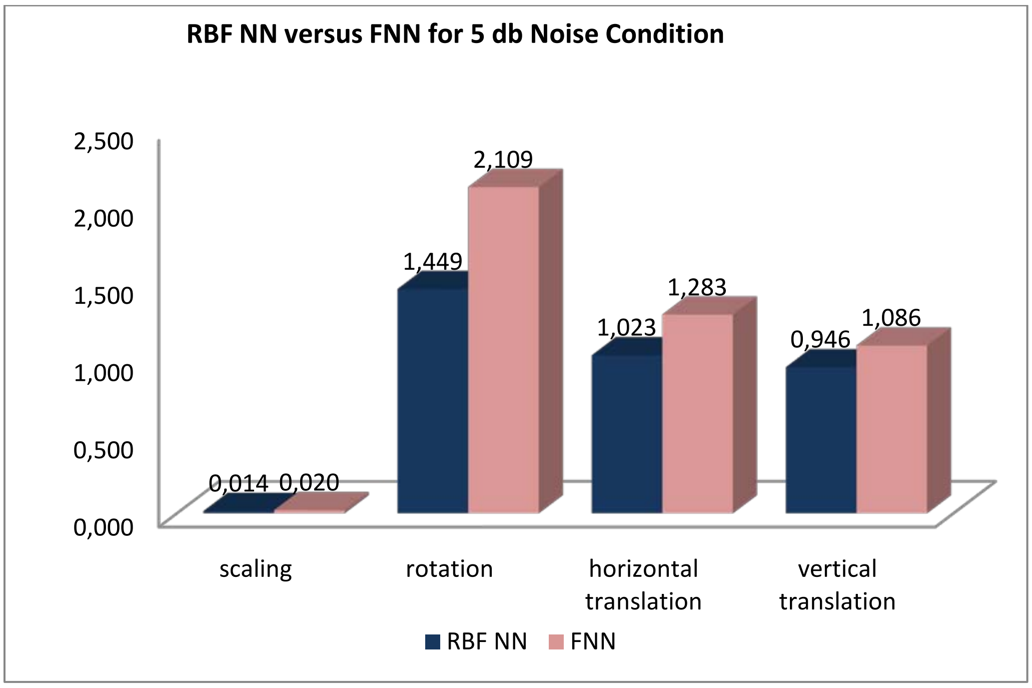

Figure 4.37 The errors of RBF NN versus FNN for 5 db noise condition for the image moon for 2DPCA features in MRCR

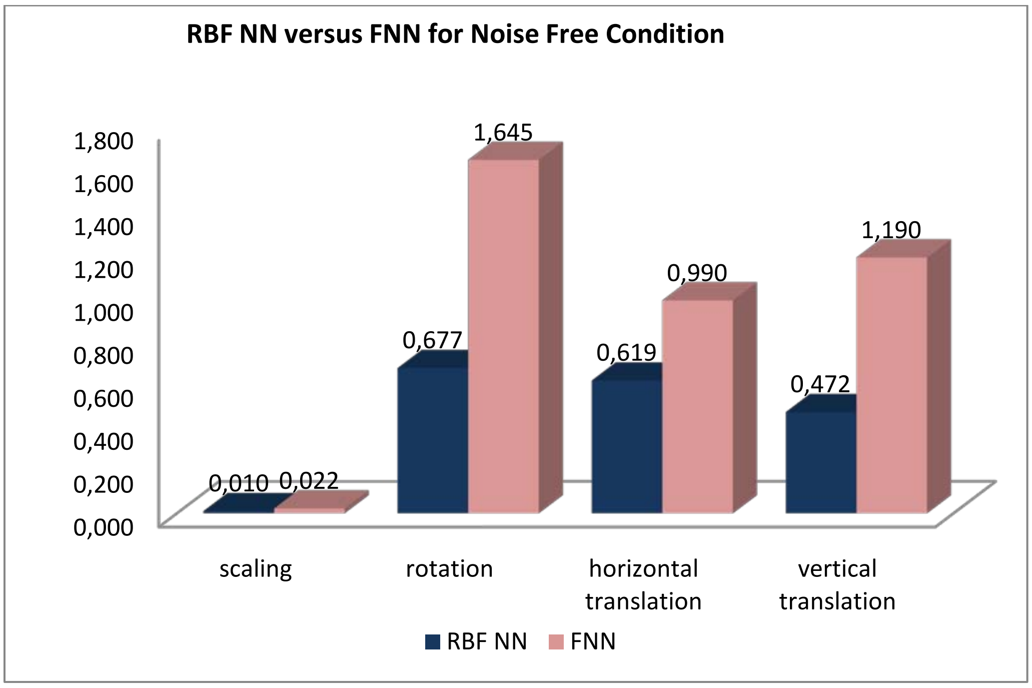

Figure 4.38 The errors of RBF NN versus FNN for noise free condition for the image girl for 2DPCA features in MRCR

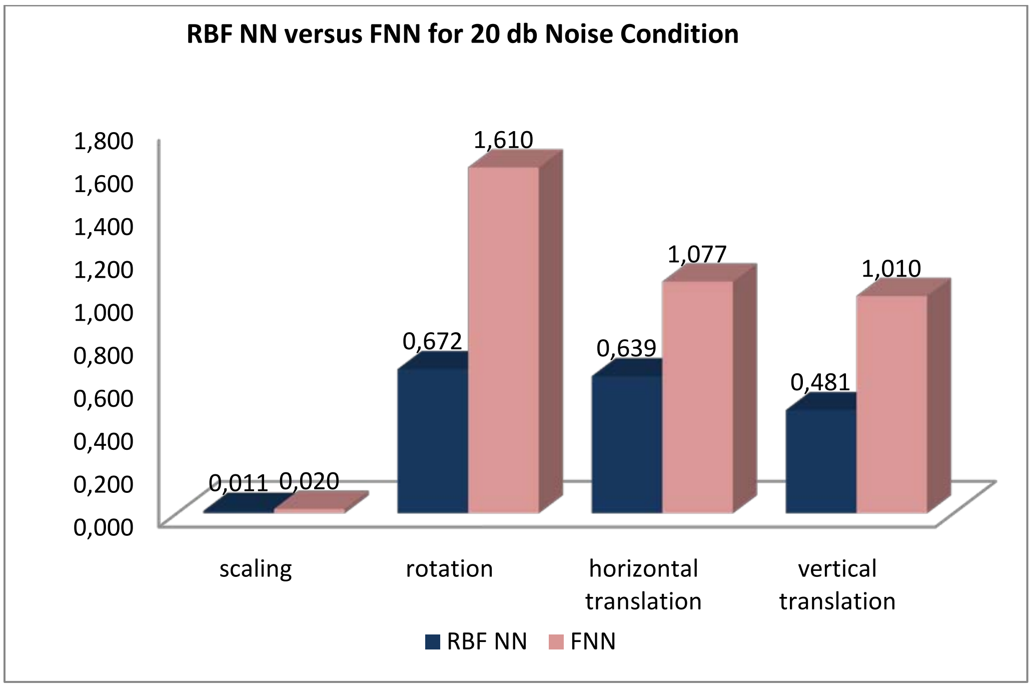

Figure 4.39 The errors of RBF NN versus FNN for 20 db condition for the image girl for 2DPCA features in MRCR

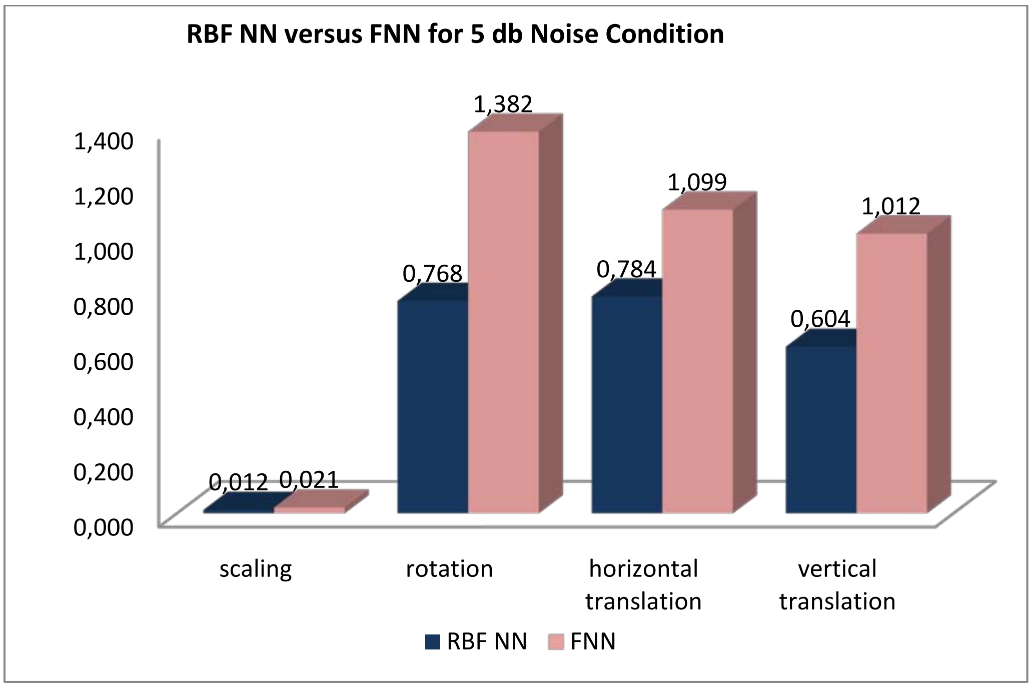

Figure 4.40 The errors of RBF NN versus FNN for 5 db condition for the image girl for 2DPCA features in MRCR

The comparison between RBFNN-based and FNN-based schemes is given in Table 4.6.

Table 4.6 Comparison between RBFNN-based and FNN-based schemes

Issues	**RBFNN-based**	**FNN-based**
Need for input/output data normalization	No	Yes
Need for network generalization methods	No	Yes
Need for multiple training trials	No	Yes
LRFR performance	Superior for high SNR	
MRCR performance	Superior	
Time for single training	< 1 sec	between 15-50 sec

4.6.3 DCT and 2D PCA Performance Comparison

In this section, the comparison of DCT and 2DPCA features is given for both local range fine registration (LRFR) and medium range coarse registration (MRCR). The graphs in the Figures from 4.41 to 4.58 refer to LRFR category and Figures from 4.59 to 4.76 refer to MRCR category. In these graphs, the performances of DCT and 2DPCA features are compared both for RBFNN and FNN. In the comparisons, it was seen that in noise free conditions, DCT has a superior performance compared to 2DPCA. However, while the level of noise increases, that is increasing the noise from 20 db to 5 db; the performance of 2DPCA first becomes equal to the performance of DCT, and when the noise is in the level of 5 db, its performance becomes a little bit better than DCT for both LRFR and MRCR modes. The graphs of the experiments are given in the Figures from 4.41 to 4.76 in the sections 4.6.3.1., 4.6.3.2., 4.6.3.3 and 4.6.3.4.

4.6.3.1 The Results of LRFR of RBF NN

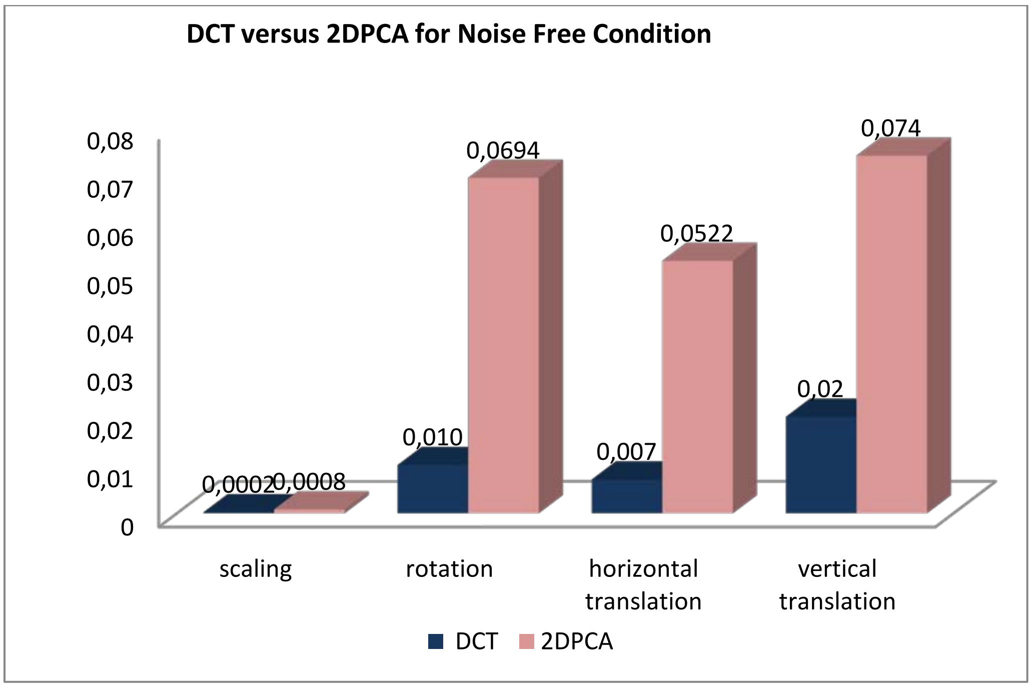

Figure 4.41 The errors of RBF NN for DCT features versus 2DPCA features for noise free condition for the image aerial in LRFR

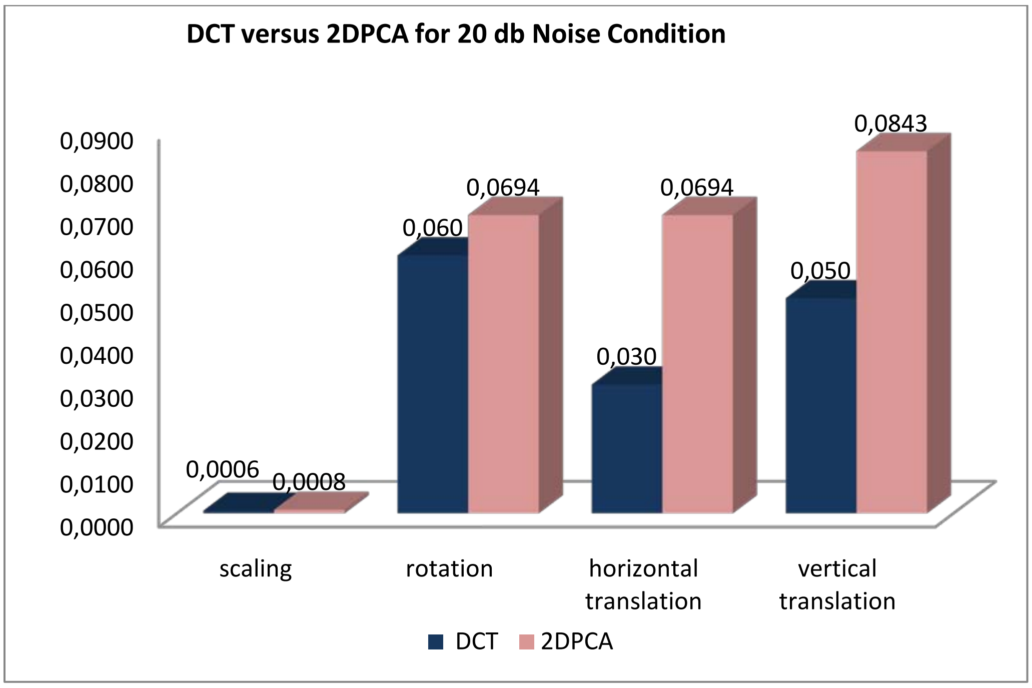

Figure 4.42 The errors of RBF NN for DCT features versus 2DPCA features for 20 db noise condition for the image aerial in LRFR

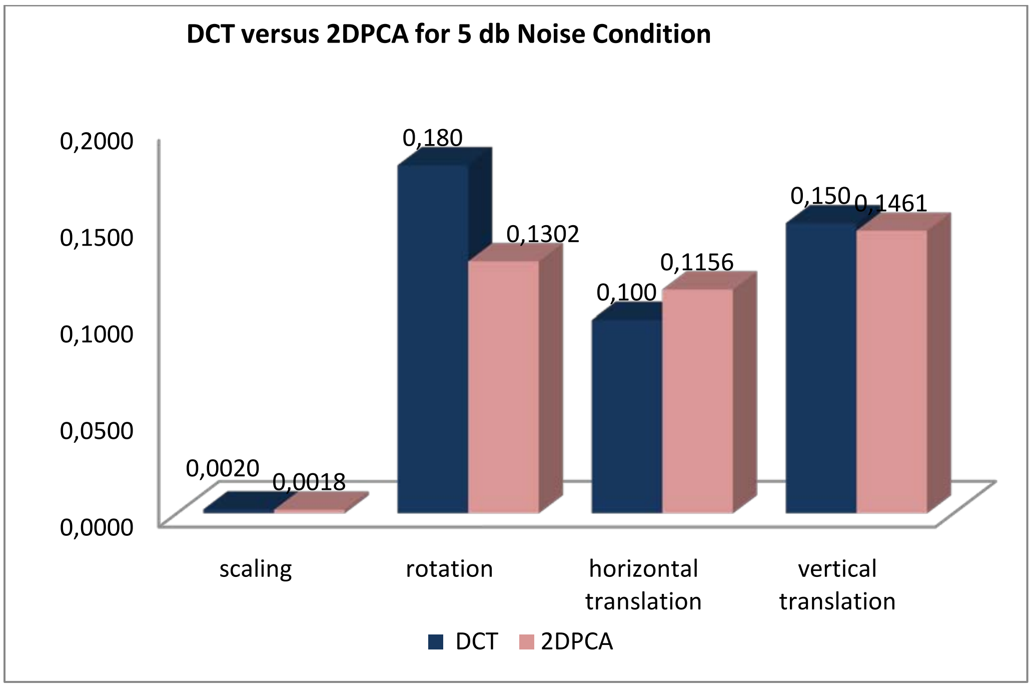

Figure 4.43 The errors of RBF NN for DCT features versus 2DPCA features for 5 db noise condition for the image aerial in LRFR

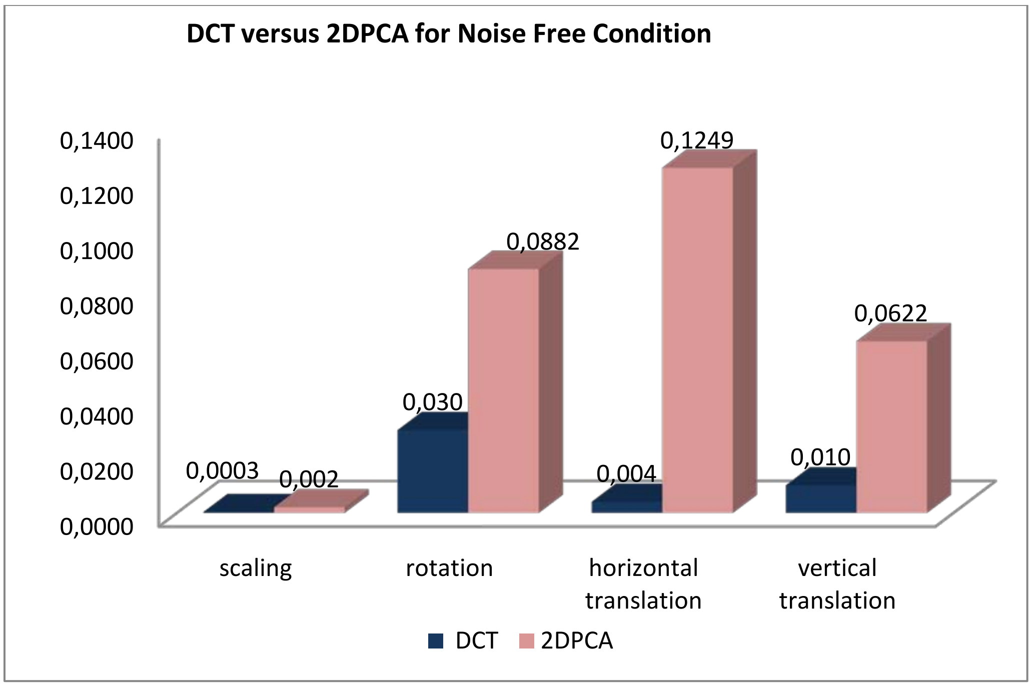

Figure 4.44 The errors of RBF NN for DCT features versus 2DPCA features for noise free condition for the image moon in LRFR

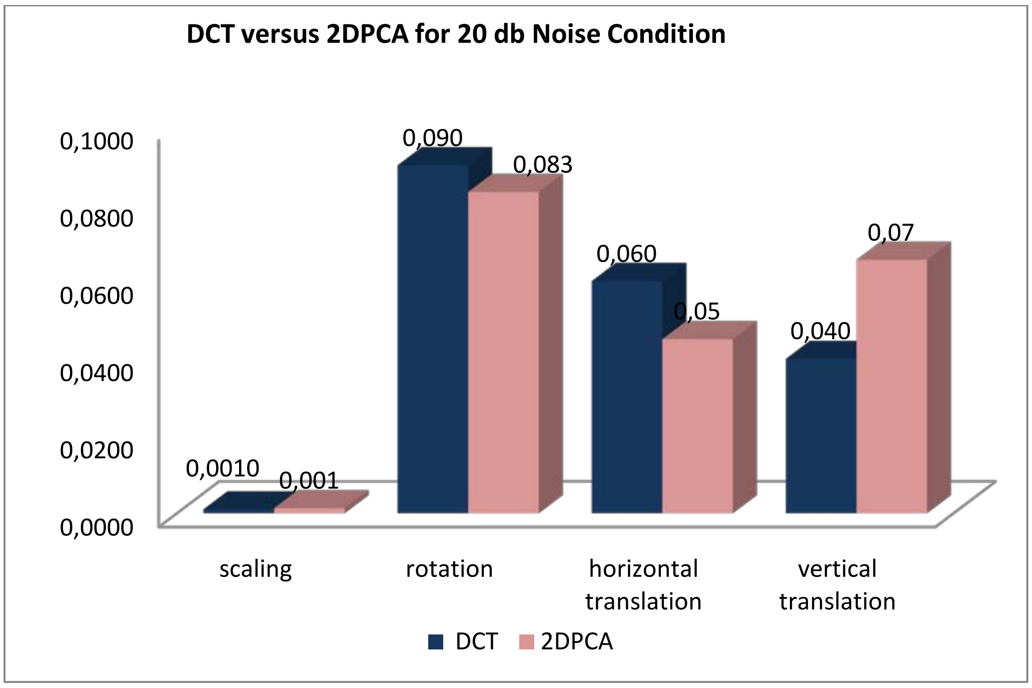

Figure 4.45 The errors of RBF NN for DCT features versus 2DPCA features for 20 db noise condition for the image moon in LRFR

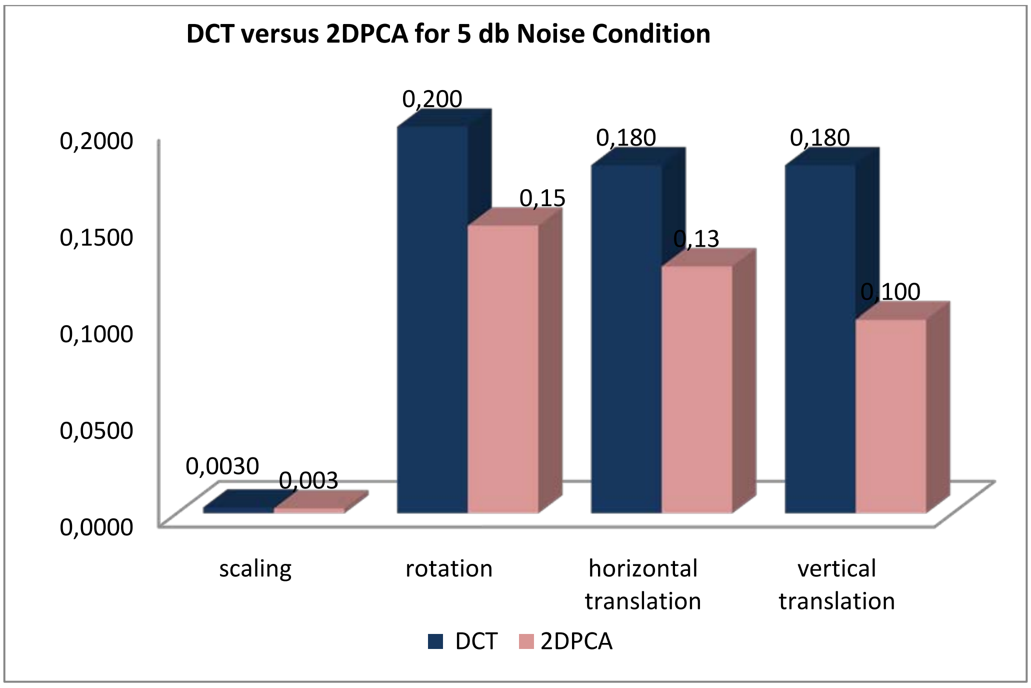

Figure 4.46 The errors of RBF NN for DCT features versus 2DPCA features for 5 db noise condition for the image moon in LRFR

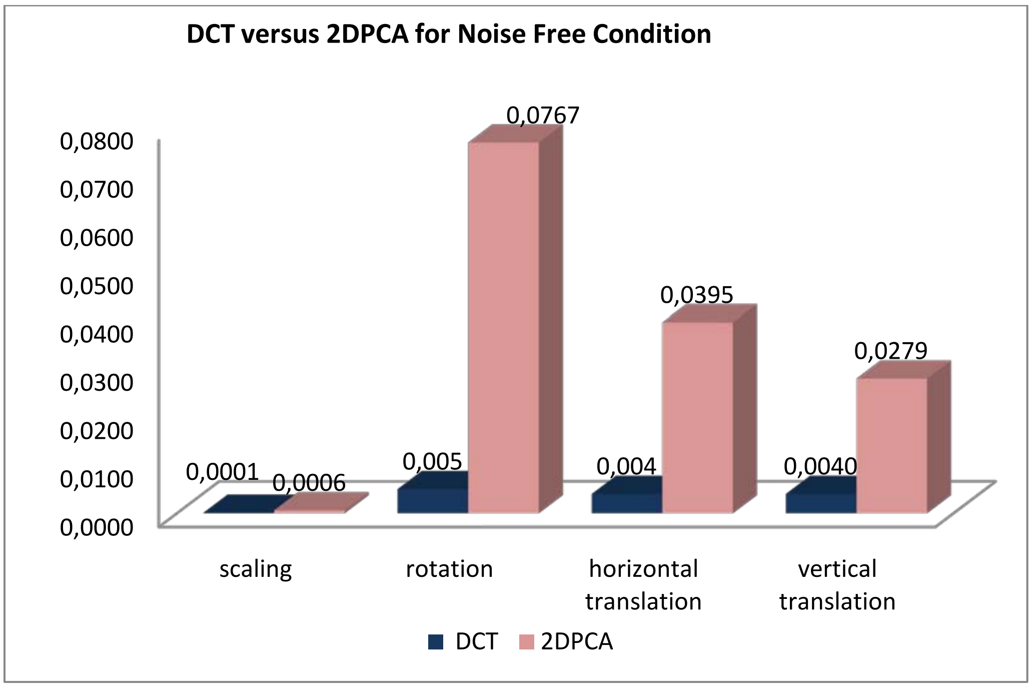

Figure 4.47 The errors of RBF NN for DCT features versus 2DPCA features for noise free condition for the image girl in LRFR

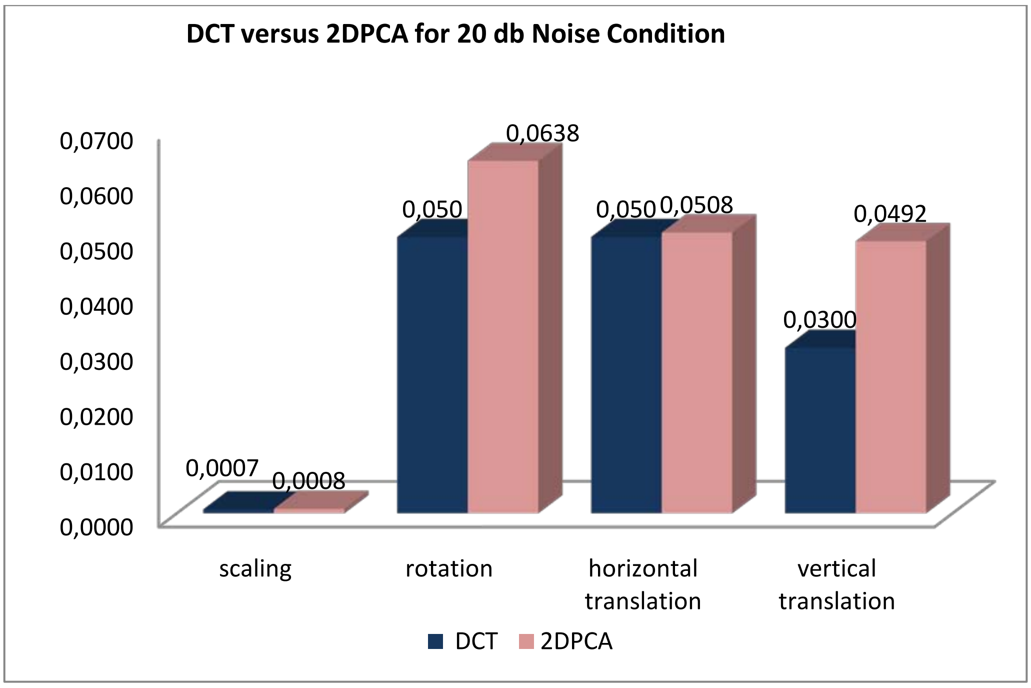

Figure 4.48 The errors of RBF NN for DCT features versus 2DPCA features for 20 db noise condition for the image girl in LRFR

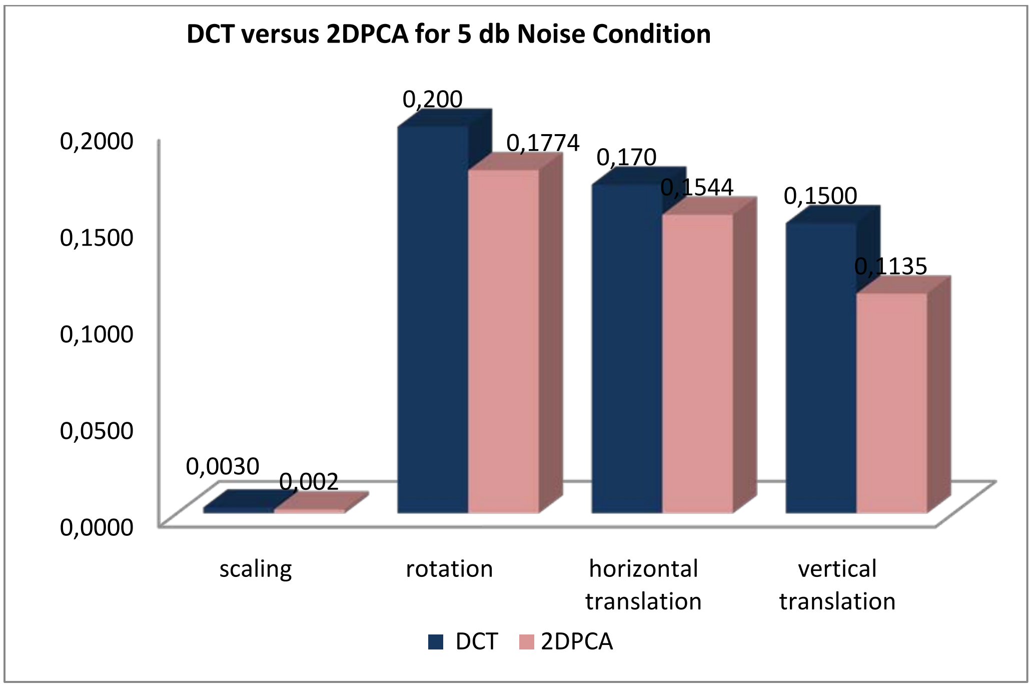

Figure 4.49 The errors of RBF NN for DCT features versus 2DPCA features for 5 db noise condition for the image girl in LRFR

4.6.3.2 The results of LRFR of FNN

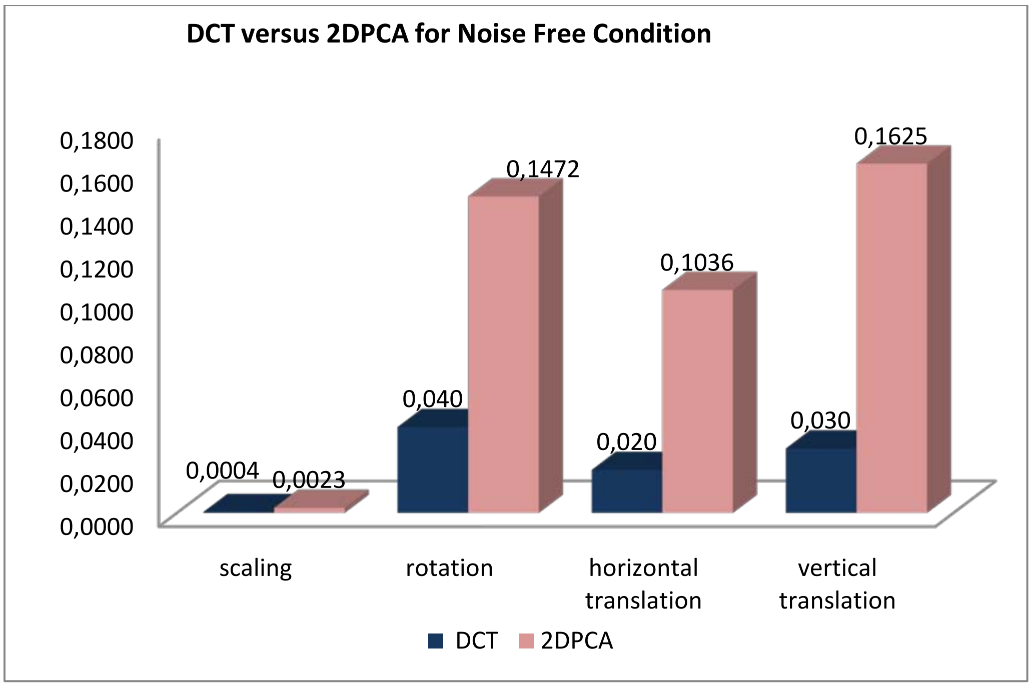

Figure 4.50 The errors of FNN for DCT features versus 2DPCA features for noise free condition for the image aerial in LRFR

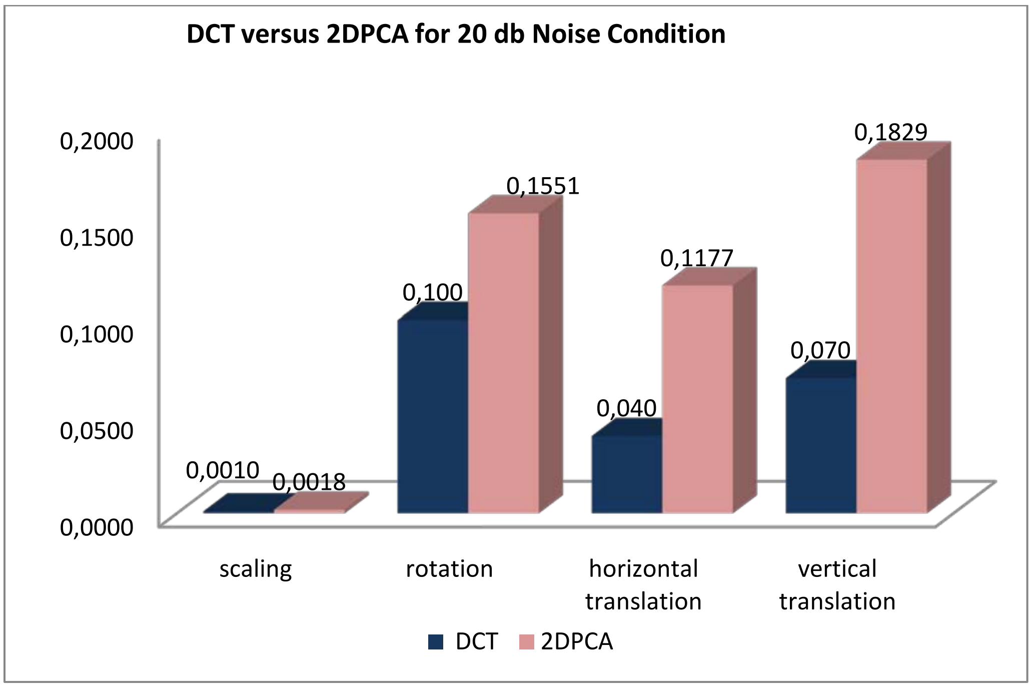

Figure 4.51 The errors of FNN for DCT features versus 2DPCA features for 20 db noise condition for the image aerial in LRFR

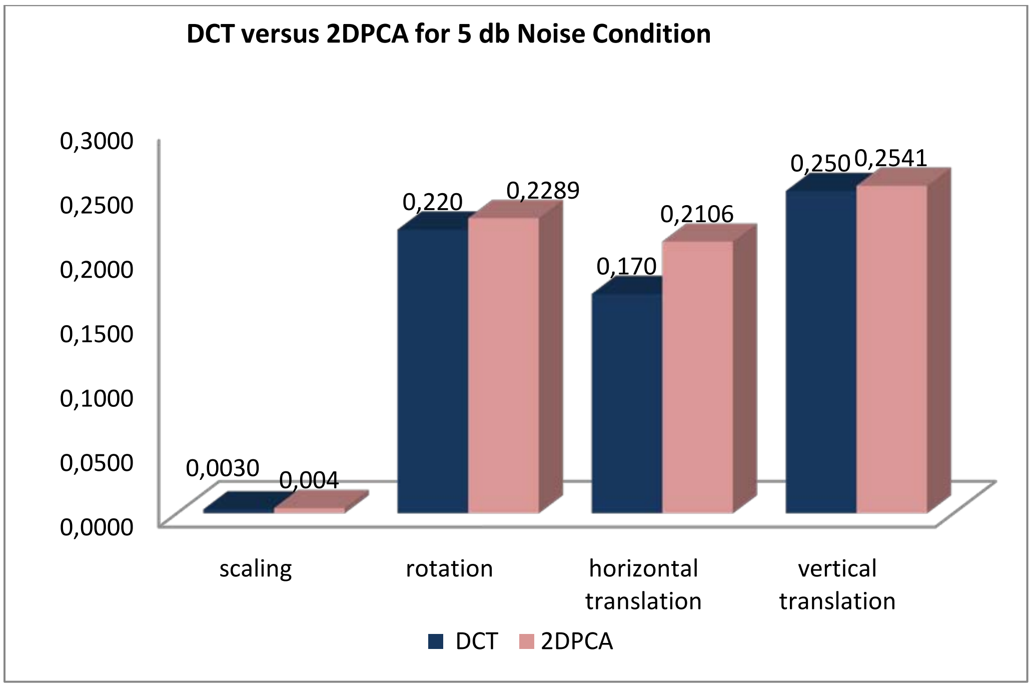

Figure 4.52 The errors of FNN for DCT features versus 2DPCA features for 5 db noise condition for the image aerial in LRFR

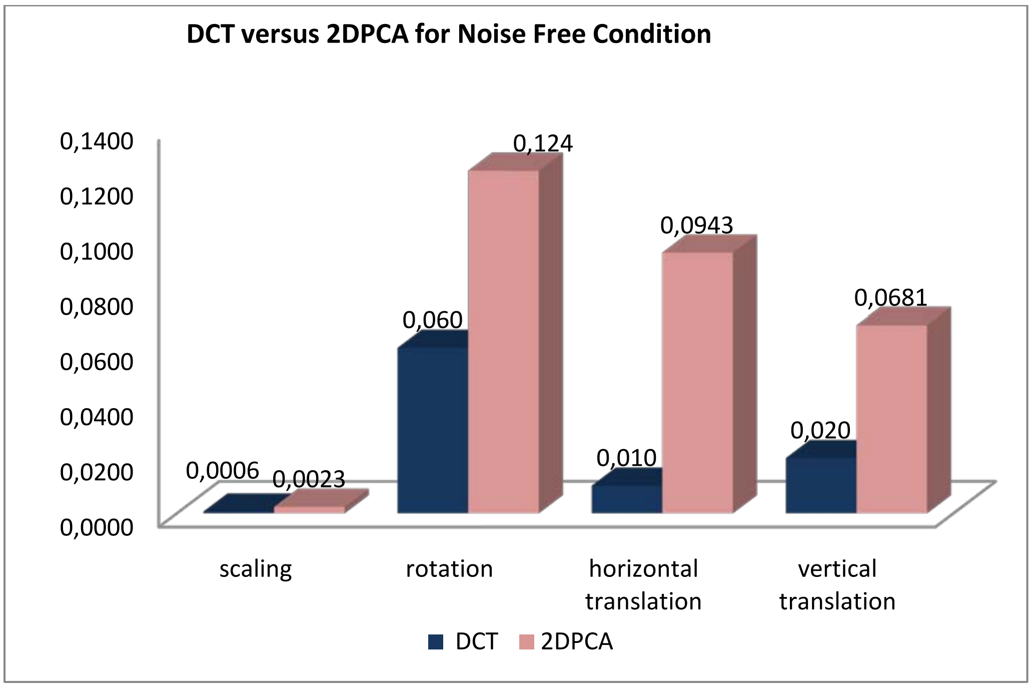

Figure 4.53 The errors of FNN for DCT features versus 2DPCA features for noise free condition for the image moon in LRFR

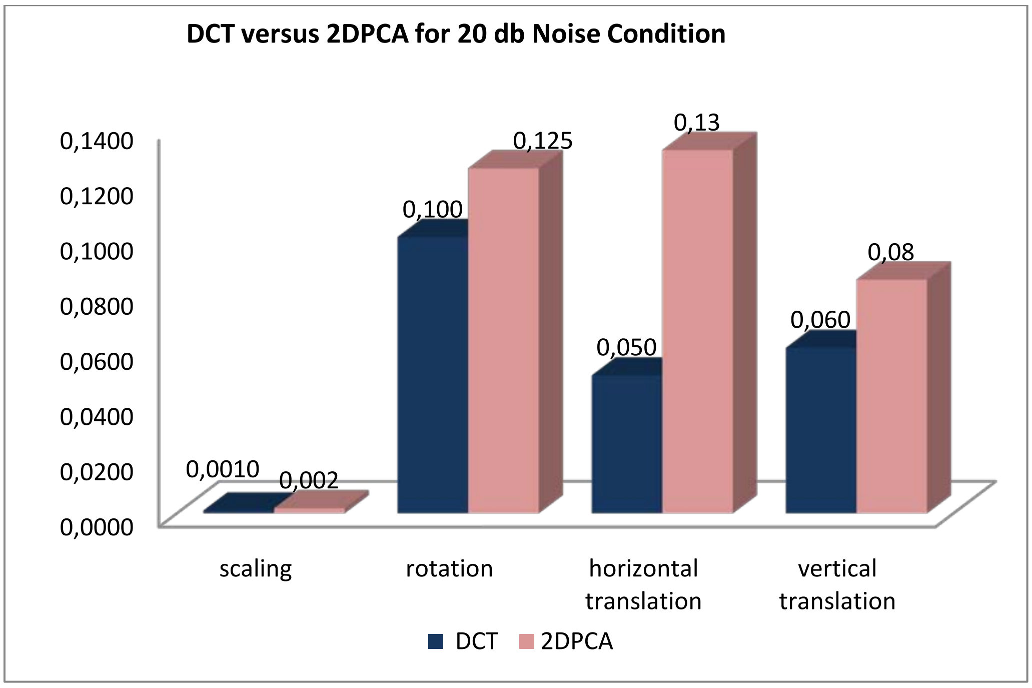

Figure 4.54 The errors of FNN for DCT features versus 2DPCA features for 20 db noise condition for the image moon in LRFR

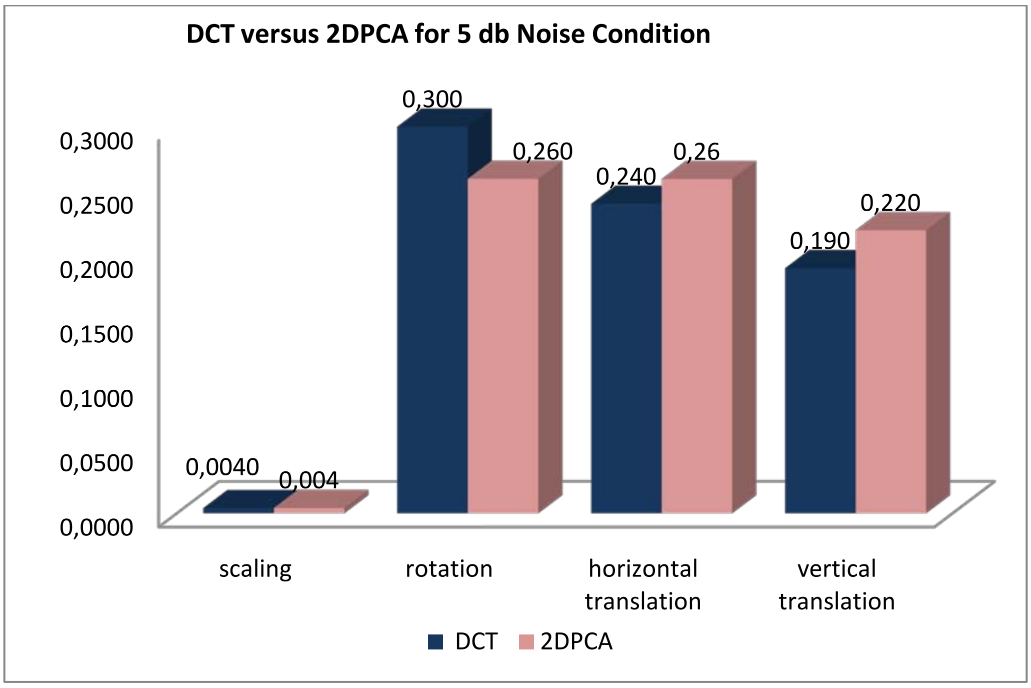

Figure 4.55 The errors of FNN for DCT features versus 2DPCA features for 5 db noise condition for the image moon in LRFR

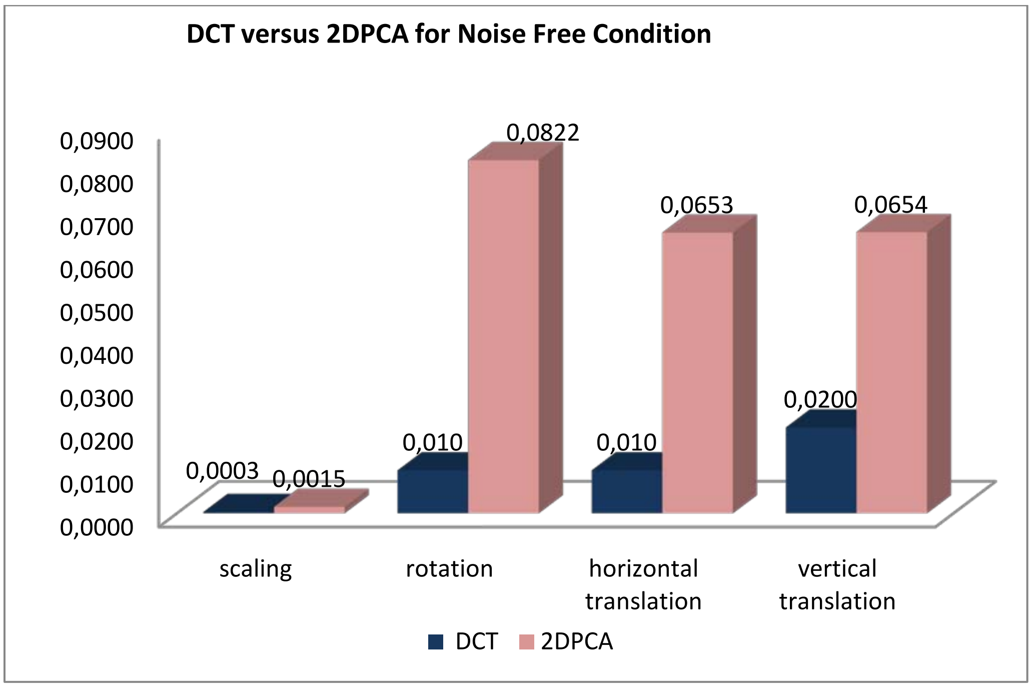

Figure 4.56 The errors of FNN for DCT features versus 2DPCA features for noise free condition for the image girl in LRFR

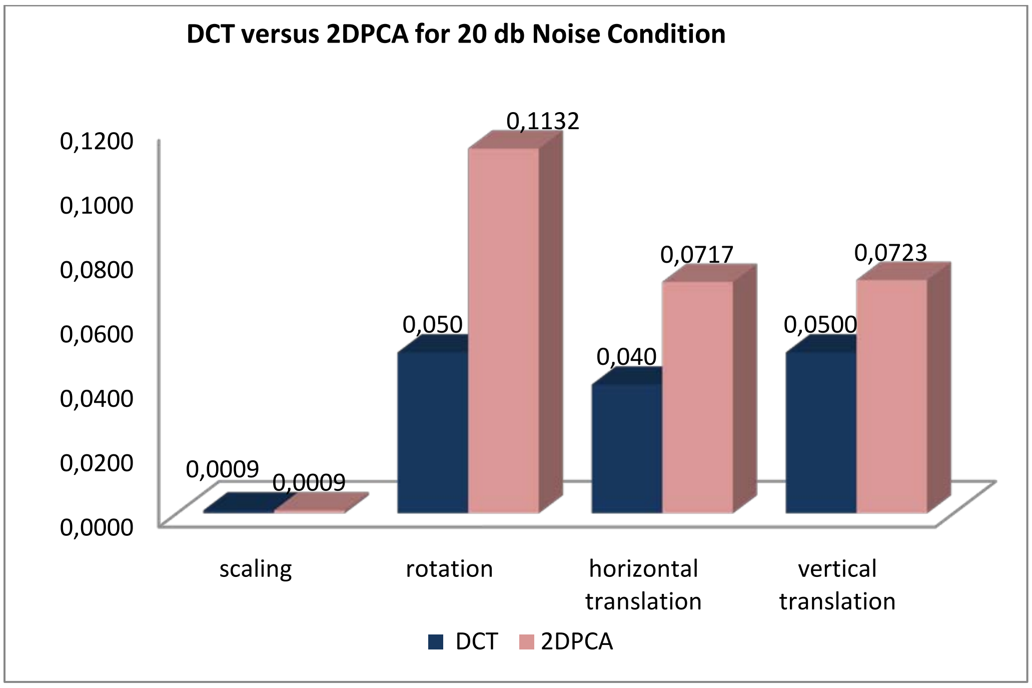

Figure 4.57 The errors of FNN for DCT features versus 2DPCA features for 20 db noise condition for the image girl in LRFR

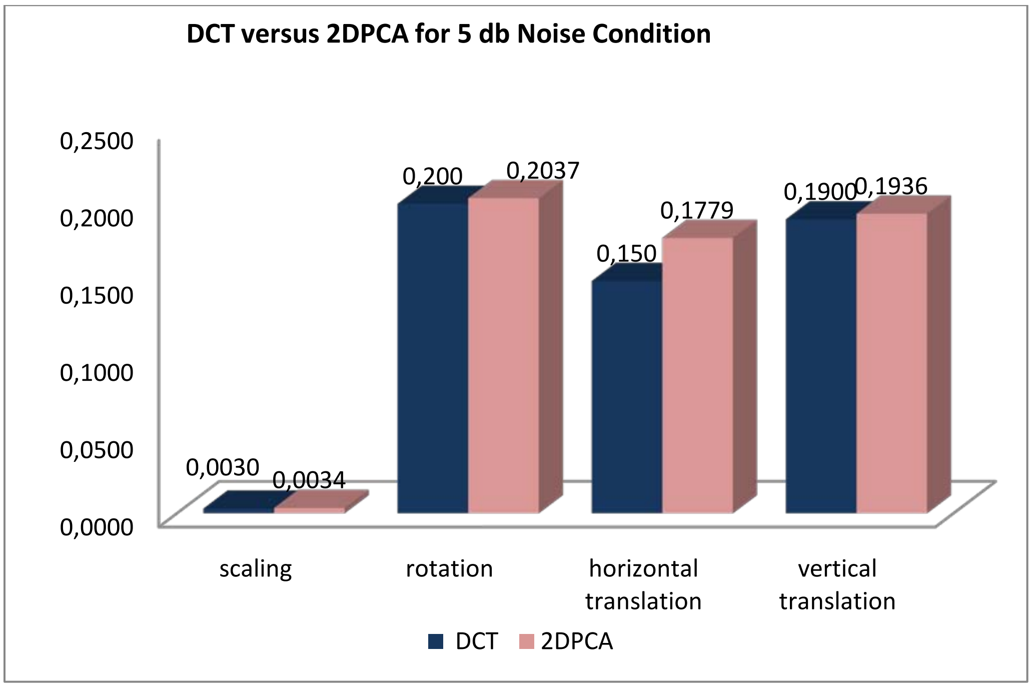

Figure 4.58 The errors of FNN for DCT features versus 2DPCA features for 5 db noise condition for the image girl in LRFR

4.6.3.3 The Results of MRCR of RBF NN

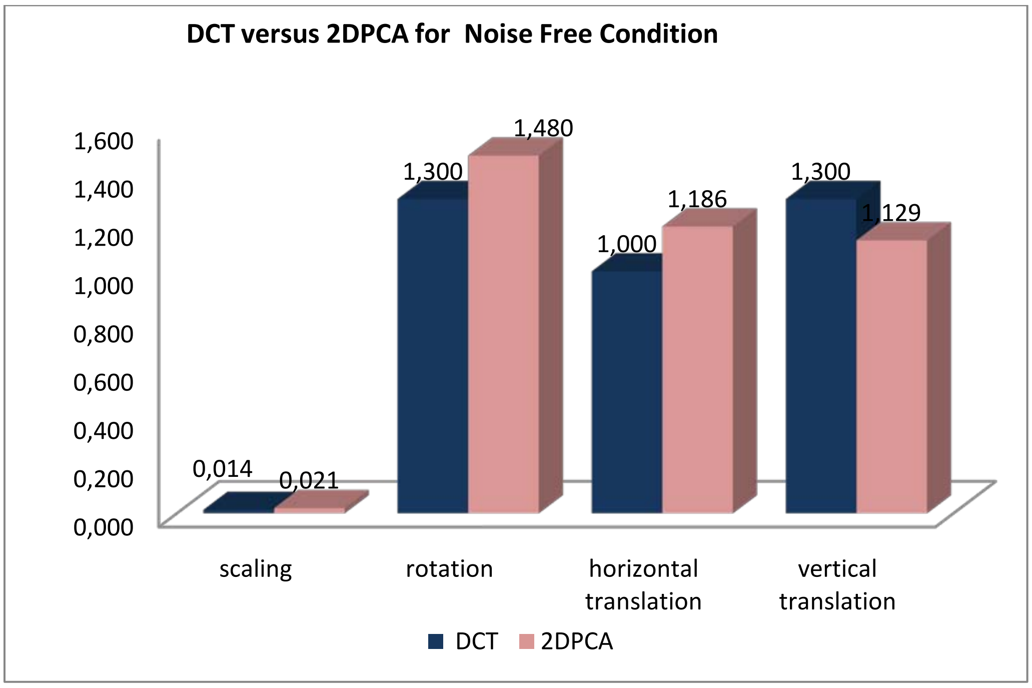

Figure 4.59 The errors of RBF NN for DCT features versus 2DPCA features for noise free condition for the image aerial in MRCR

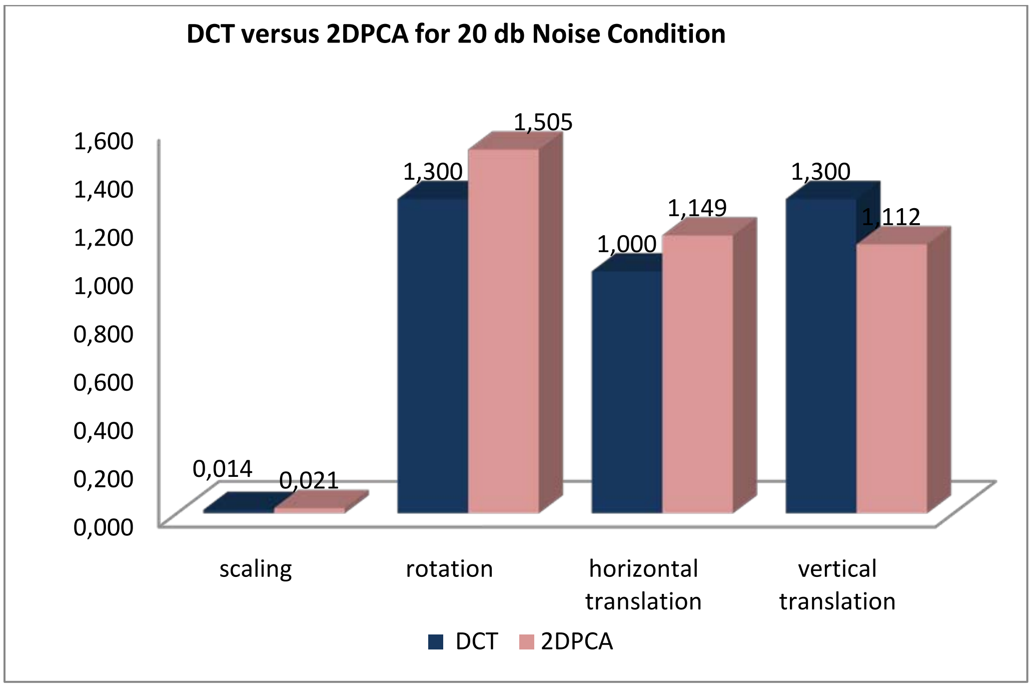

Figure 4.60 The errors of RBF NN for DCT features versus 2DPCA features for 20 db noise condition for the image aerial in MRCR

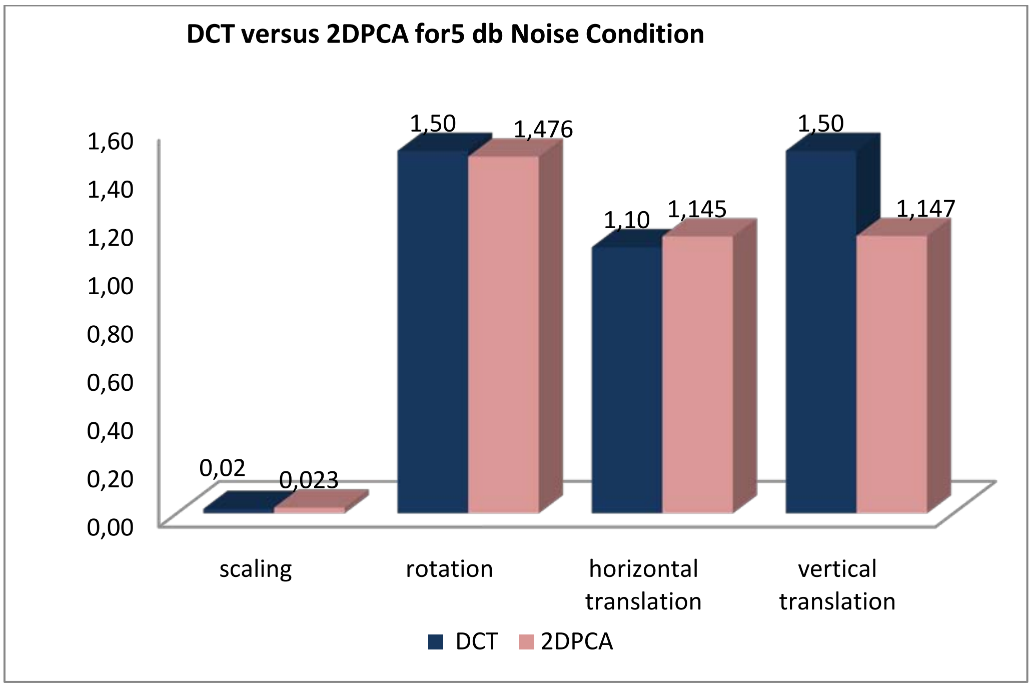

Figure 4.61 The errors of RBF NN for DCT features versus 2DPCA features for 5 db noise condition for the image aerial in MRCR

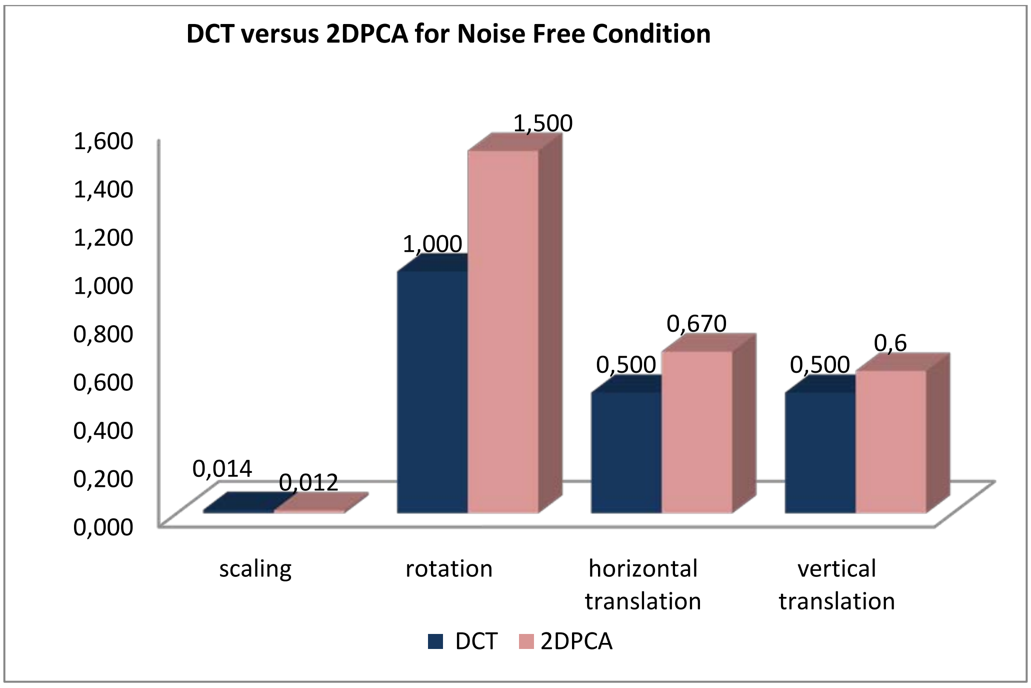

Figure 4.62 The errors of RBF NN for DCT features versus 2DPCA features for noise free condition for the image moon in MRCR

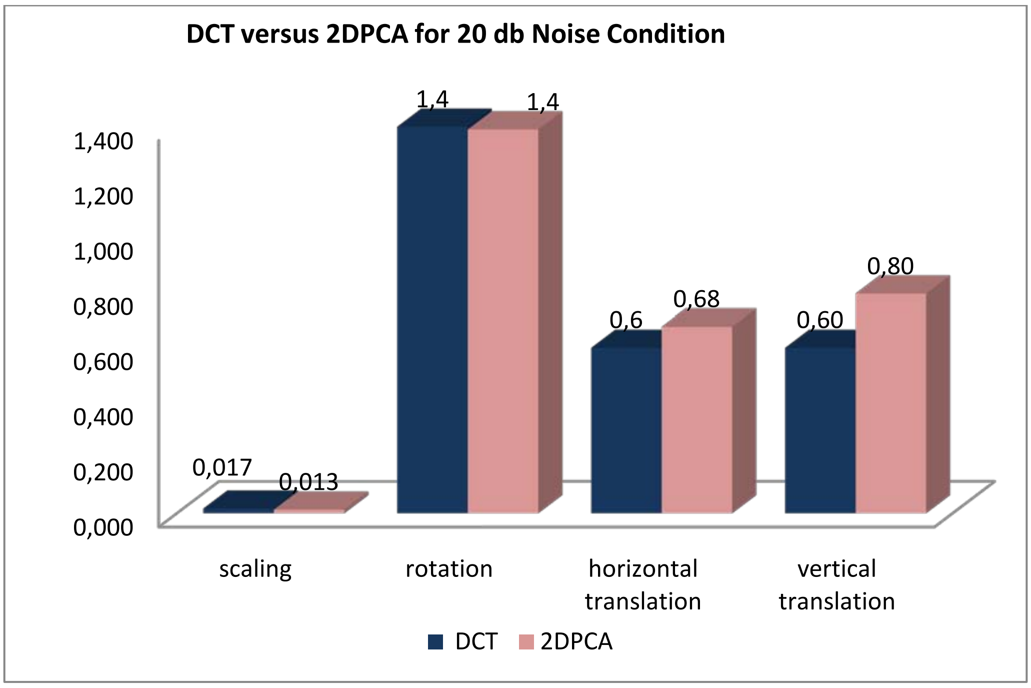

Figure 4.63 The errors of RBF NN for DCT features versus 2DPCA features for 20 db noise condition for the image moon in MRCR

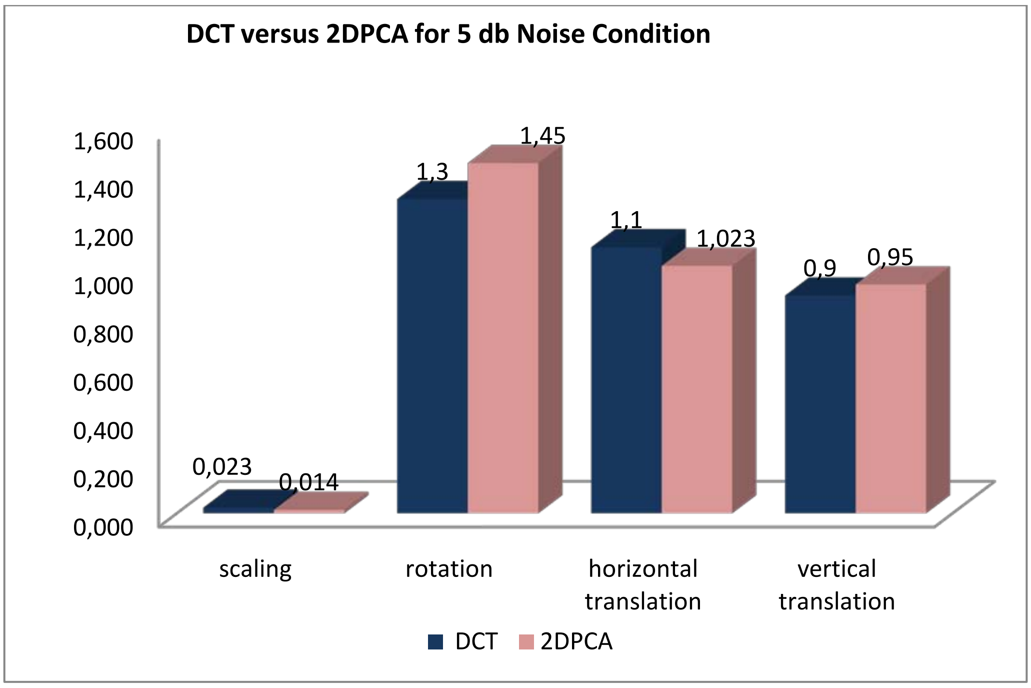

Figure 4.64 The errors of RBF NN for DCT features versus 2DPCA features for 5 db noise condition for the image moon in MRCR

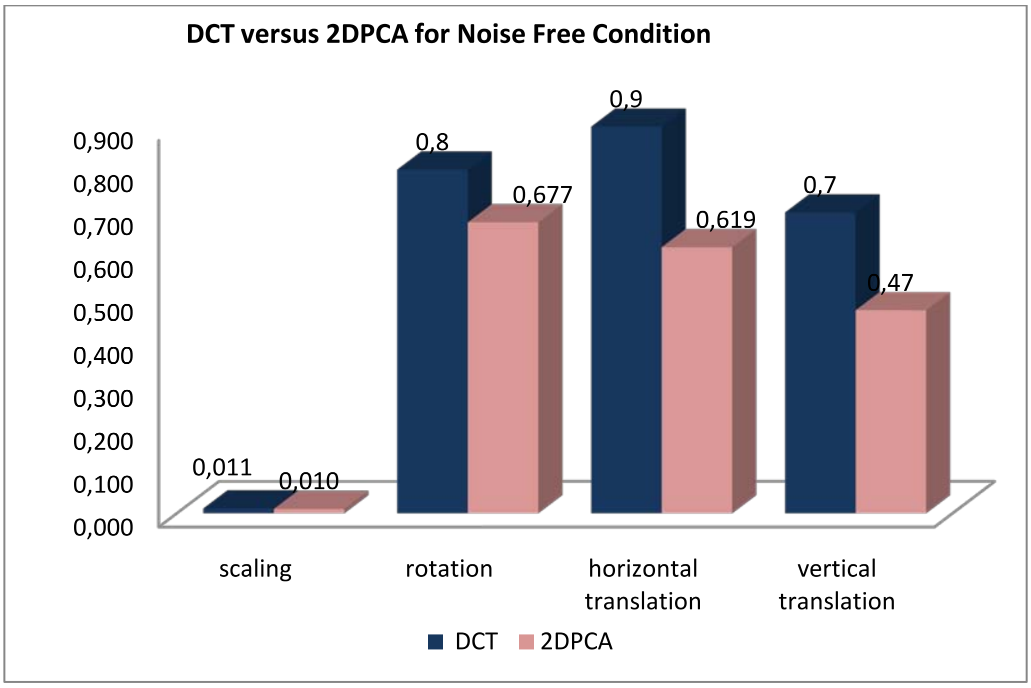

Figure 4.65 The errors of RBF NN for DCT features versus 2DPCA features for noise free condition for the image girl in MRCR

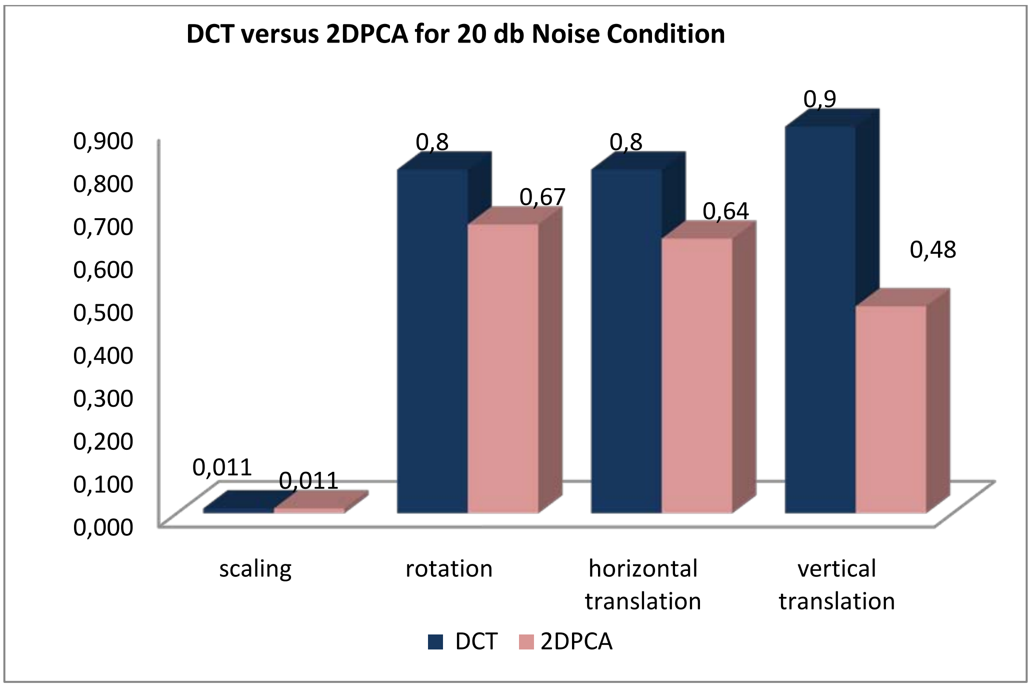

Figure 4.66 The errors of RBF NN for DCT features versus 2DPCA features for 20 db noise condition for the image girl in MRCR

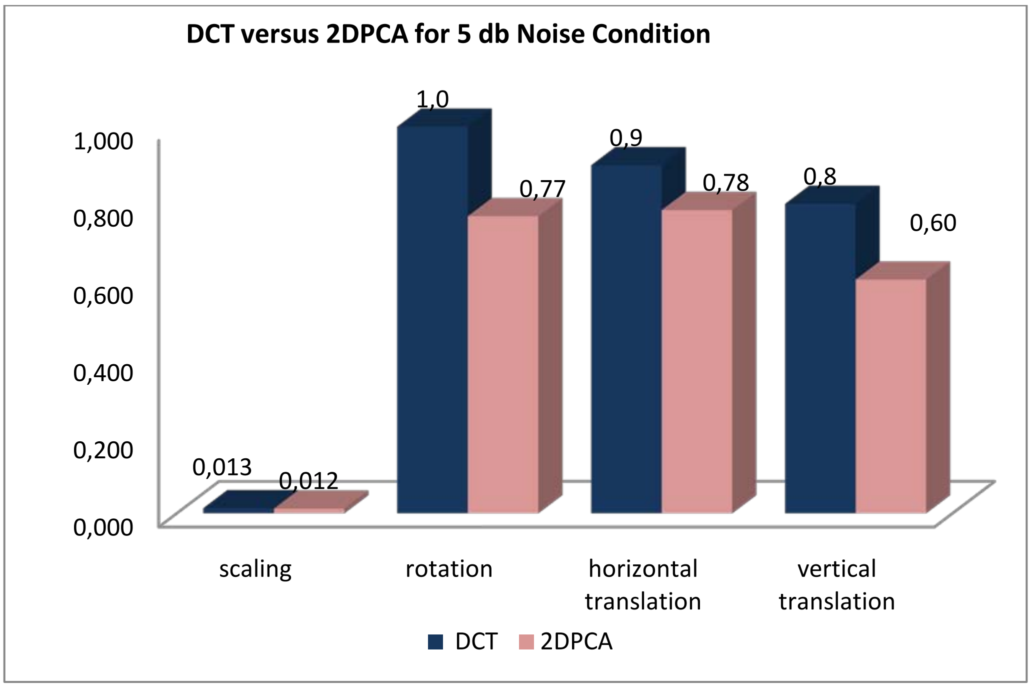

Figure 4.67 The errors of RBF NN for DCT features versus 2DPCA features for 5 db noise condition for the image girl in MRCR

4.6.3.4 The Results of MRCR of FNN

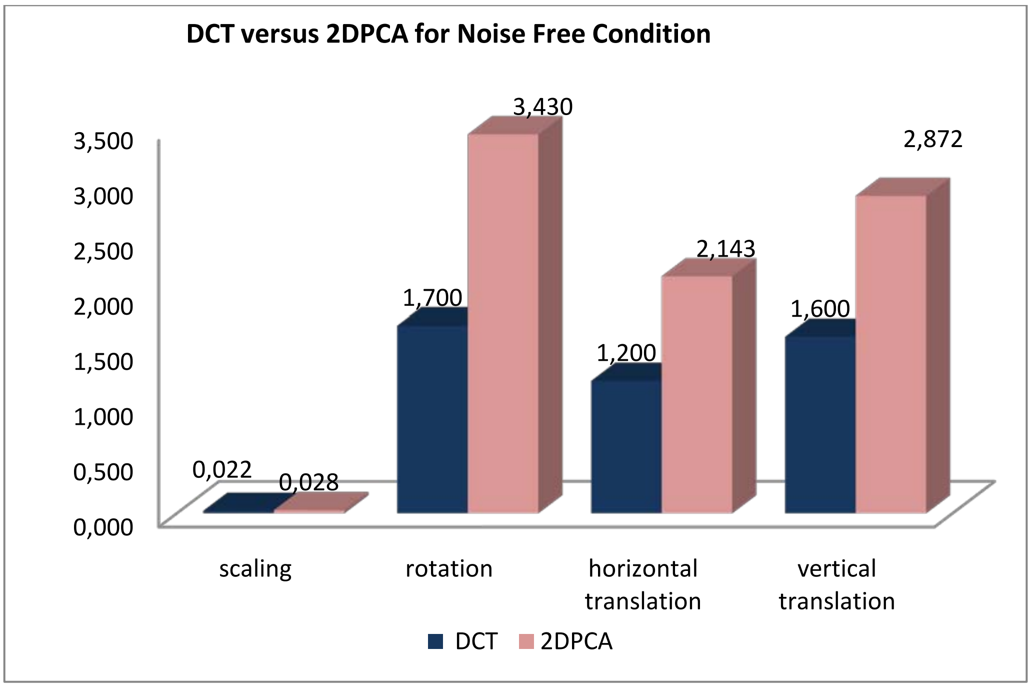

Figure 4.68 The errors of FNN for DCT features versus 2DPCA features for noise free condition for the image aerial in MRCR

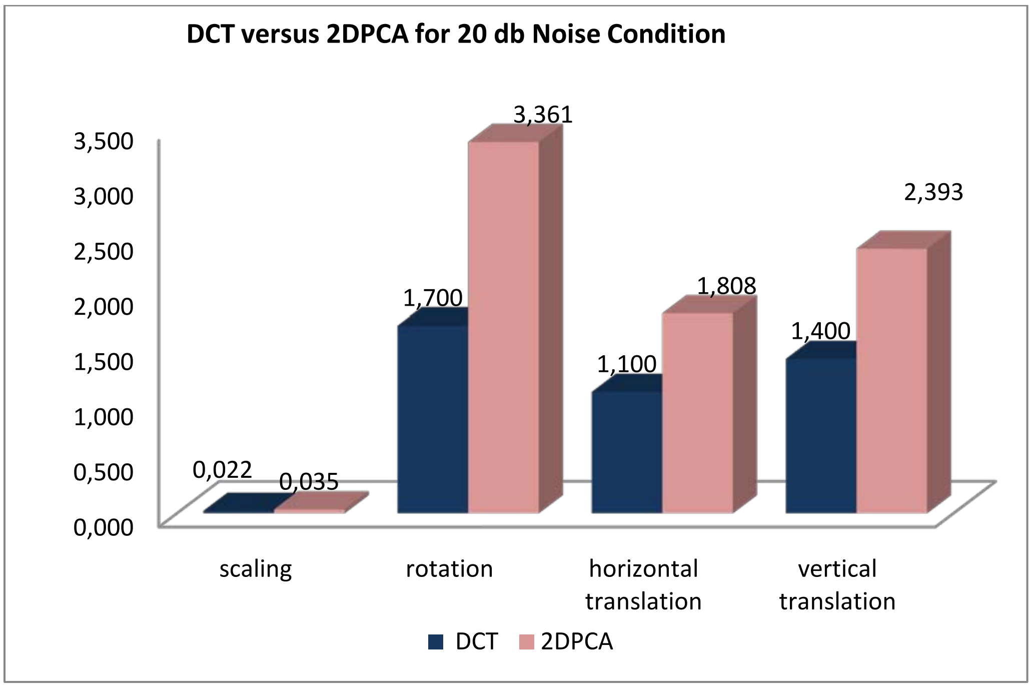

Figure 4.69 The errors of FNN for DCT features versus 2DPCA features for 20 db noise condition for the image aerial in MRCR

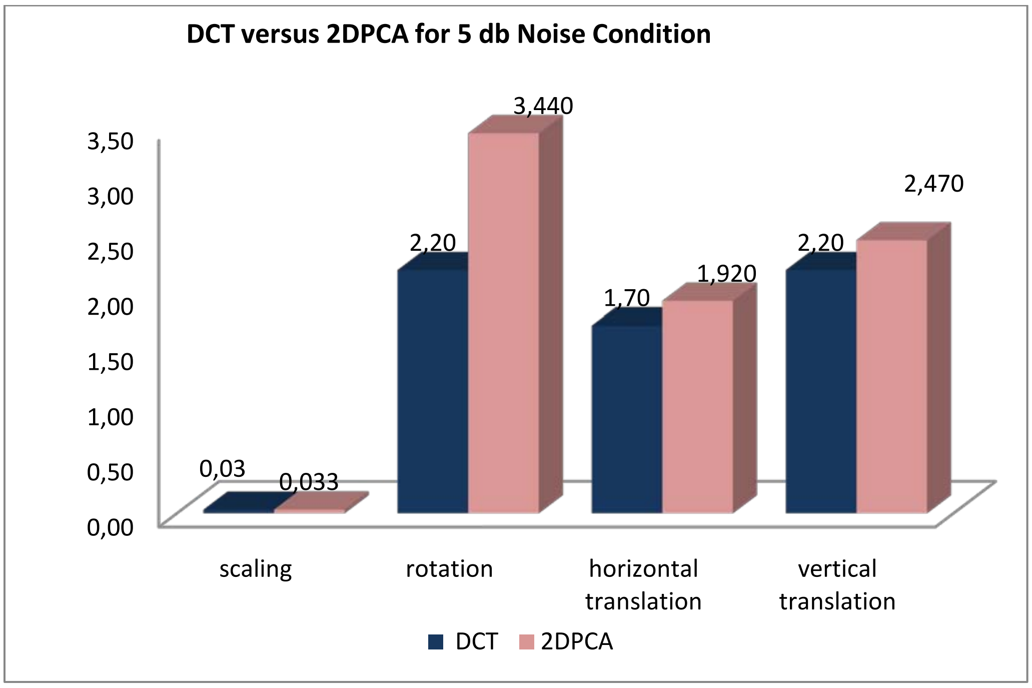

Figure 4.70 The errors of FNN for DCT features versus 2DPCA features for 5 db noise condition for the image aerial in MRCR

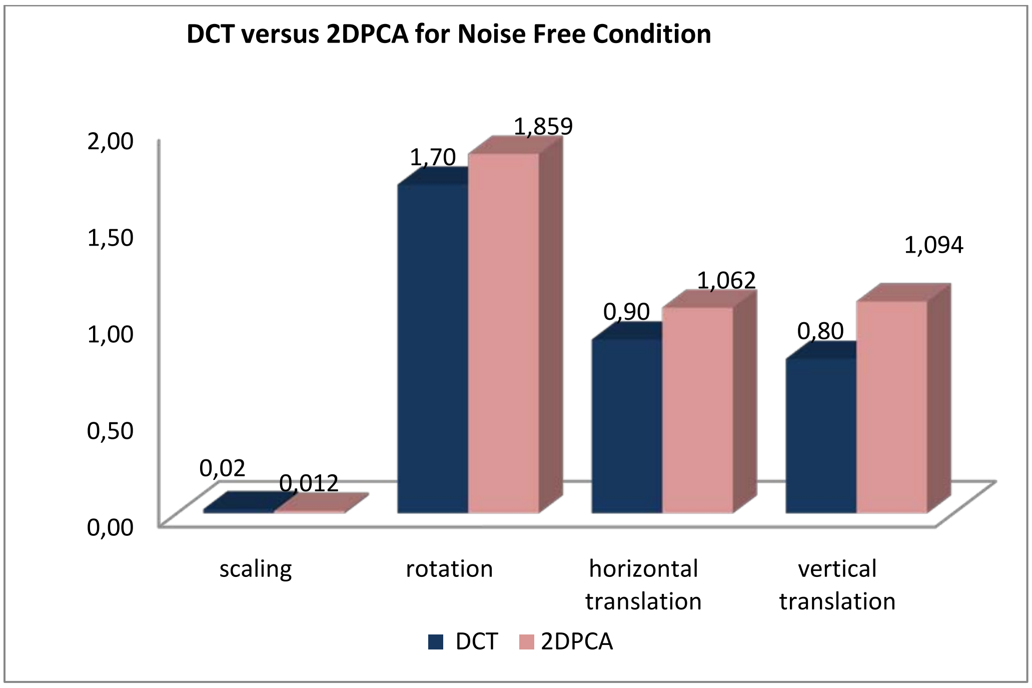

Figure 4.71 The errors of FNN for DCT features versus 2DPCA features for noise free condition for the image moon in MRCR

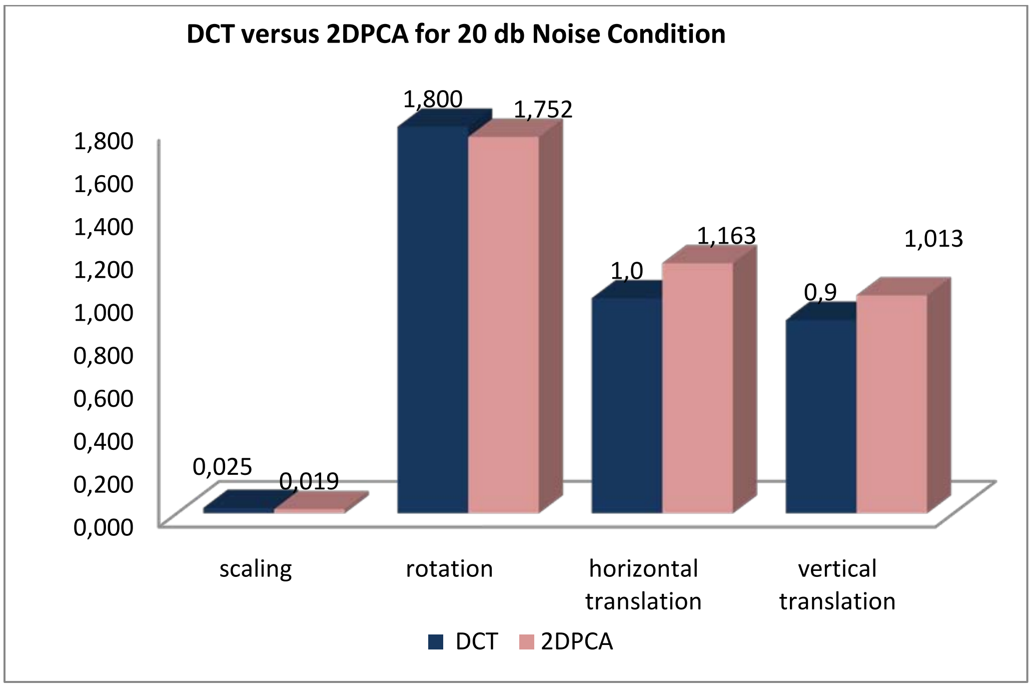

Figure 4.72 The errors of FNN for DCT features versus 2DPCA features for 20 db noise condition for the image moon in MRCR

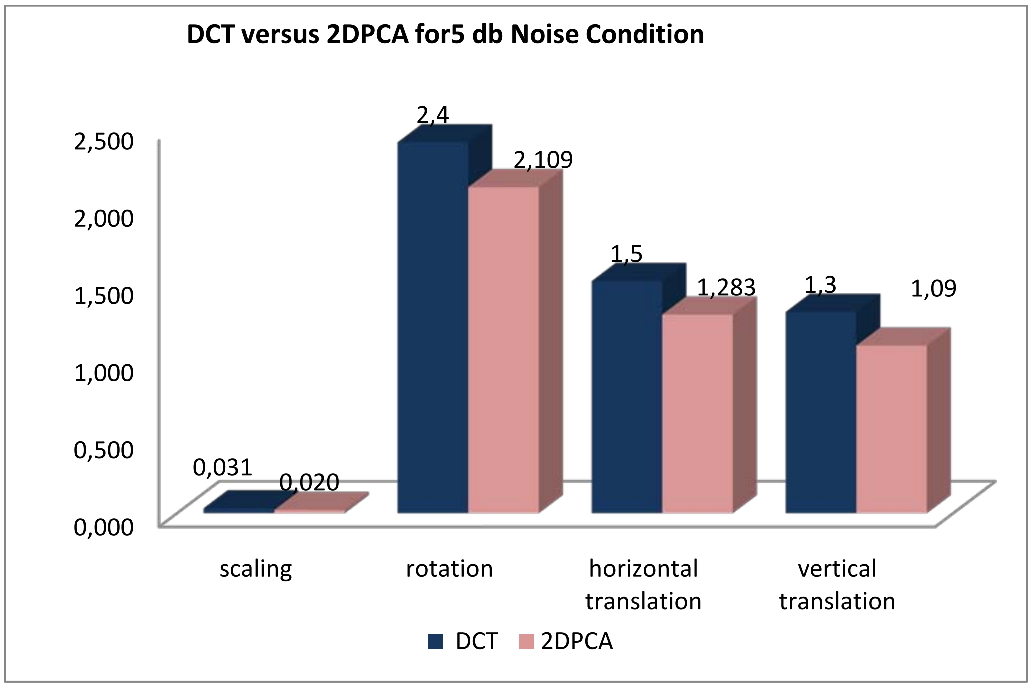

Figure 4.73 The errors of FNN for DCT features versus 2DPCA features for 5 db noise condition for the image moon in MRCR

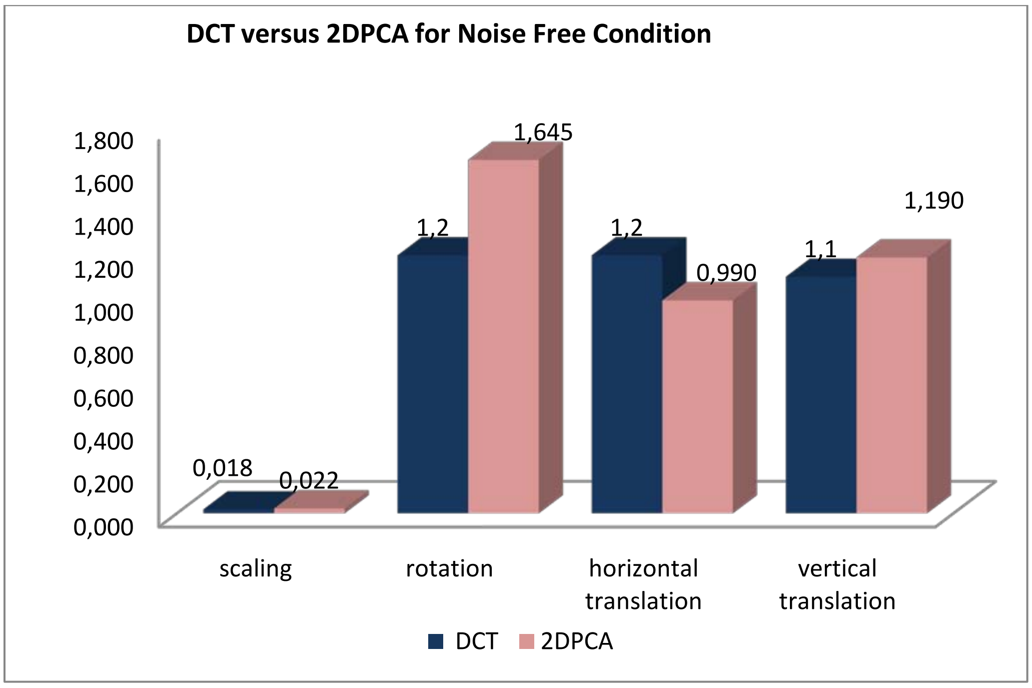

Figure 4.74 The errors of FNN for DCT features versus 2DPCA features for noise free condition for the image girl in MRCR

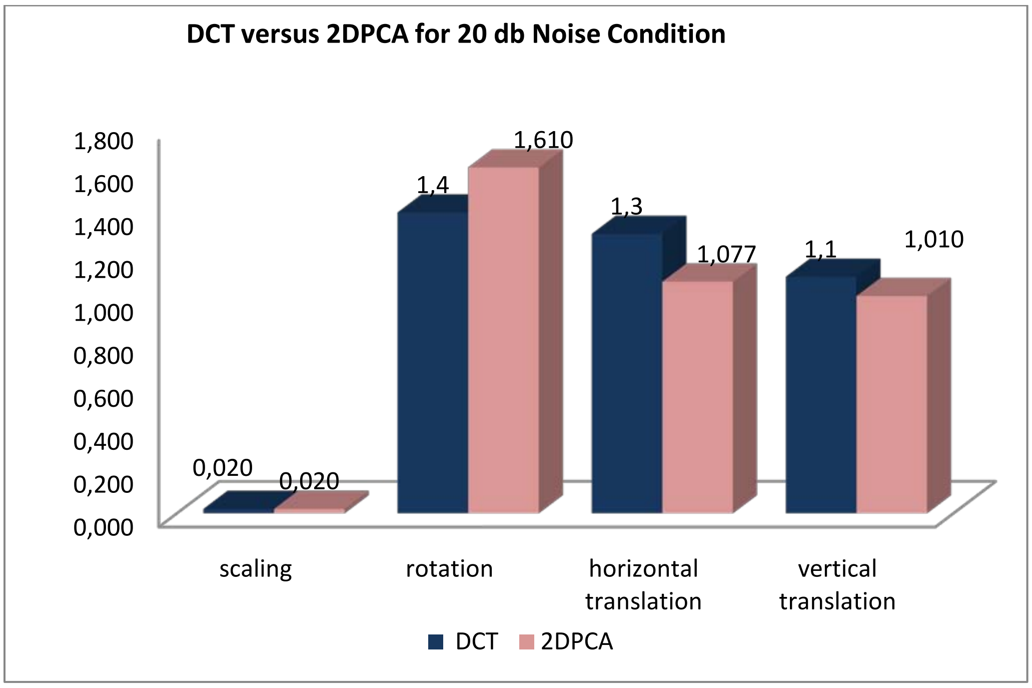

Figure 4.75 The errors of FNN for DCT features versus 2DPCA features for 20 db noise condition for the image girl in MRCR

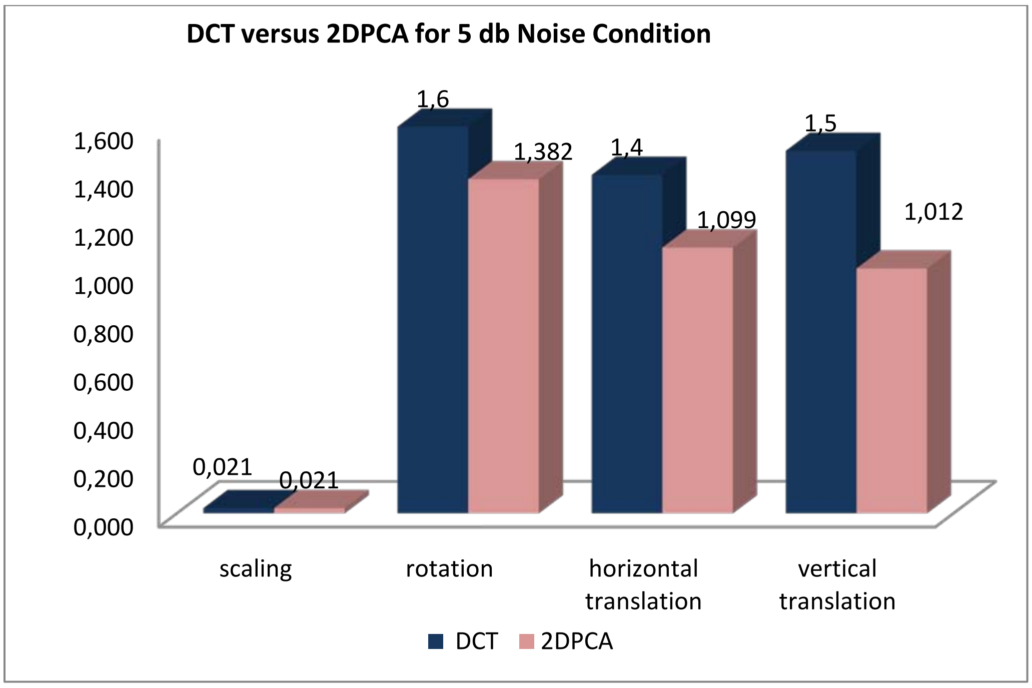

Figure 4.76 The errors of FNN for DCT features versus 2DPCA features for 5 db noise condition for the image girl in MRCR

CHAPTER FIVE

CONCLUSION

In this thesis, neural network based image registration process was released in the presence of noise. Two different types of neural networks were used for registration, which are RBFNN and FNN. Both of these networks are fed by global image features such as DCT coefficients and two dimensional PCA. The results of RBFNN versus FNN networks and also DCT versus 2DPCA coefficients are compared in the experimental work part with the help of graphs and tables.

In all the experiments done, it was seen that employing a RBFNN instead of FNN to estimate affine registration parameters gives more accurate results in the presence of noise. In addition to this, generally the proposed scheme shows good robustness to noise in general. The other important advantage of using RBFNN instead of FNN-based scheme is its fastness and ease of implementation. By this way we avoid the disadvantages of FNN-based scheme such as lengthy training iterations, multiple training attempts and network generalization problem which are encountered during the training stage. In FNN, there are lots of parameters such as number of hidden layers, number of hidden layer neurons, etc. In order to determine these parameters, we have to do lots of tries. However, in RBFNN the only parameter has to be determined is the spread parameter of the Gaussian Function. Although the optimal spread value changes due to the changes in the training data in the training stage, the experiments show that any suboptimal spread value can be easily estimated and used without decreasing the performance drastically.

The proposed scheme was applied both with DCT features and 2DPCA features. In the experiments, it was seen that DCT features gives more accurate results in noise free situation. However, in the presence of noise, 2DPCA has a better performance. This means that, DCT features are more sensitive to the presence of noise than 2DPCA features. That is 2DPCA is more robust in the presence of noise. On the other hand, DCT has much more accurate results in the noise-free conditions or higher SNR values. As a result, DCT is much more sensitive to noise but it has higher accuracy in the noiseless conditions.

REFERENCES

Brown, L.G. (1992). A Survey Of Image Registration Techniques. *ACM Computing Surveys 24*(4), 325-376

Capel, D., & Zisserman, A. (2003). Computer vision applied to super-resolution. *IEEE Signal Process 20*(3), 75-86

Davoodi-Bojd, E., & Soltanian-Zadeh, H. (2008). Grid based registration of diffusion tensor images using least square support vector machines. *Advances in Computer Science and Engineering*. 621–628.

Elhanany, I., Sheinfeld, M., & Beck, A. (2000). Robust image registration based on feedforward neural networks. *IEEE international conference on systems, man and cybernetics 2,* 1507-1511.

Ham, F. M. & Kostanic, I. (2001). *Principles of neurocomputing for science and engineering*, Singapore: Mcgraw Hill International Edition.

Haykin, S. (1999). *Neural networks a comprehensive foundation* (2nd edition). United States of America: Prentice Hall International, Inc.

Peng, D. Q., Liu, J., Tian, J. W. & Zheng, S. (2006). Transformation model estimation of image registration via least square support vector machines. *Pattern Recognition Letters 27*(12), 1397–1404.

Qian, Z., & Li, J. (1997). Use of hopfield neural network for complex image registration. *9th international conference on tools with artificial Intelligence*. 204-207

Wachowiak, M. P., Smolikova.,R., Zurada, J. M., & Elmaghraby, A. S. (2002). A supervised learning approach to landmark-based elastic biomedical image registration and interpolation. *IEEE international joint conference on Neural Networks*, 1625-1630

Wu, J., & Xie, J. (2004). Zernike moment-based image registration scheme utilizing feedforward neural networks. *Fifth world congress on intelligent control and automation.* 4046-4048.

Xu, A., Jin, X., & Guo, P. (2006). Two-dimensional PCA combined with PCA for neural network based image registration. *International Conference on Natural Computation*, 696 – 705.

Xu, A., Jin, X., Guo, P., Bie, R. (2006). KICA feature extraction in application to FNN based image registration. *International Joint Conference on Neural Networks.* 3602-3608

Zitová, B., & Flusser, J. (2003). Image registration methods: a survey. *Image and Vision Computing 21*(11), 977-1000