Automatic Generation of RBF Networks [1]

Shayan Mukherjee [2] and Shree K. Nayar

CUCS-001-95

Department of Computer Science
Columbia University
New York, NY 10027 USA

sayan@cs.columbia.edu
nayar@cs.columbia.edu

November, 1994

[1]This research was conducted at the Center for Research in Intelligent Systems, Department of Computer Science, Columbia University. It was supported in part by the David and Lucile Packard Fellowship and in part by ARPA Contract No. DACA 76-92-C-0007.

[2]Department of Applied Mathematics and Physics, Columbia University, New York, N.Y. 10027

Abstract

Learning can be viewed as mapping from an input space to an output space. Examples of these mappings are used to construct a continuous function that approximates the given data and generalizes for intermediate instances. Radial basis function (RBF) networks are used to formulate this approximating function. A novel method is introduced that automatically constructs a RBF network for a given mapping and error bound. This network is shown to be the smallest network within the error bound for the given mapping. The integral wavelet transform is used to determine the parameters of the network. Simple one-dimensional examples are used to demonstrate how the network constructed using the transform is superior to one constructed using standard ad hoc optimization techniques. The paper concludes with the automatic generation of a network for a multidimensional problem, namely, object recognition and pose estimation. The results of this application are favorable.

1 Introduction

Learning can be thought of as mapping values from an input space to an output space or constructing a smooth function that performs the mapping. This function is constructed from a set of example mappings, namely, the training data. Since the function is smooth, it interpolates the known data, that is, it generalizes information about the mapping from the known examples. In this context, learning is equivalent to generating a continuous function that approximates the given data. A rigorous formulation of this approximating function results in a weighted sum of radial basis functions (RBF) [12]. Therefore, the approximating function can easily be cast into a neural network called RBF networks [12]. These networks have been shown to be universal approximators, theoretically capable of approximating any function to a reasonable degree of precision [1] with only one layer of basis functions. Networks with sigmoidal basis functions are universal approximators as well, however, they require two layers of basis functions. The rigorous formulation of the RBF network and the fact that it consists of only one layer of bases makes it appealing for a variety of practical applications.

One condition imposed in the formulation of RBF networks is that the approximating function consists of as many basis functions as training examples, which in practical applications is typically large. This leads to large networks that are inefficient in both speed and memory. For this reason, fewer basis functions than examples are generally used. The positions of these basis functions, the number of basis functions, the spans of the basis functions, and their weights are all parameters of the network. Setting these parameters turns into a problem of nonlinear optimization. Typically, a programmer selects the number of basis functions in an arbitrary manner and then optimizes the remaining network parameters. If the error functional of this network is not within the desired error bound, the number of basis functions is increased and the procedure is repeated. In addition, since nonlinear optimization is involved, good initial estimates of the network parameters are needed to avoid local minima in the parameter space. Hence, the rigor of RBF networks is greatly sacrificed during implementation.

The fundamental issue raised in this paper is the following. Given an error bound and a set of training examples how many basis functions are required and what are the parameters of the bases (positions, spans, and weights)? An analytic solution to this problem is proposed and an algorithm is introduced that constructs the smallest network that satisfies the error bound. This network has many attractive properties. The most significant is that the number of basis functions are found analytically rather than the ad hoc approach of adding more and more bases to the approximating function and repeating the nonlinear optimization until the error bound is satisfied.

A transform technique is used to determine the number of basis functions required and other network parameters. If the basis functions were sines and cosines, the Fourier transform could be used determine the parameters of the bases. Parseval's theorem can then be used to calculate the number of basis functions required for high performance. A similar approach is employed here to set the parameters of the RBF network. Instead of the Fourier transform, the integral wavelet transform is used [5]. The basis functions for the integral wavelet transform are orthonormal and local. These local orthonormal bases can be constructed from the set of radial basis functions. Once the basis function is chosen, the integral wavelet transform is applied to the training data. The number of wavelets required to achieve a certain error bound is found using Parseval's theorem. Once the coefficients of the wavelets are calculated, the wavelets and their coefficients are directly mapped to a RBF network.

The benefits of using the wavelet transform to determine network parameters are both time and space. The off-line time required to determine network parameters is greatly decreased because once the transform coefficients are calculated the exact number of basis functions required is known. This is in contrast to performing a nonlinear optimization using more and more basis functions until the desired error bound is achieved. The on-line time required for the network to perform the mapping is also decreased since the generated network is the smallest one for a given error bound. For the same reason, the amount of memory required to store the network is also minimized.

The transform approach and the conventional approach to setting network parameters are compared for two 1-dimensional functions. The convergence of the error functional, as the number of basis functions increases, is examined for both approaches. The accuracy of the mappings using both approaches is also studied. The advantage of the transform approach is quite apparent from these simulations. Next, the transform approach was applied to a high dimensional problem in computer vision, namely, recognizing an object and estimating its pose. The result is a network that maps the projection of an input image in an n-dimensional subspace, called the eigenspace [11], to an object number and object pose. The effectiveness of our approach in terms of time to determine network parameters, time to perform the mapping, accuracy of the mapping, and memory required to store the network are examined. The network's performance is shown to be favorable.

The paper is organized as follows. The formulation of an input-output mapping as a RBF network is described in section 2. The relation between an integral wavelet transform and a RBF network is presented in section 3. The computational complexity and memory requirements of this mapping with respect to the dimensionality of the input and output spaces and number of basis functions used is discussed in section 4. Section 5 presents a simple example that demonstrates the advantage of the transform approach. Finally, the transform approach is applied to the problem of object recognition and pose estimation. The paper is concluded with a discussion of a variety of issues related to the proposed scheme.

2 Radial Basis Function Networks

The strength of RBF networks is that input-output mappings are learned in a mathematically rigorous formalism. At the heart of all neural-network schemes is the question of whether a multivariate function can be represented exactly by sums and products of univariate functions. In the case of RBF networks, this representation is formulated using approximation theory and regularization techniques. How RBF networks relate to the problem of input-output mapping, the formulation of a network, and short-comings in this formulation are described in this section. The first part of this section is a summary of Poggio and Girosi [12].

Learning a mapping between an input and output space is often viewed as determining a function that performs the mapping. It can be posed as the problem of approximating a continuous multivariate function $f(\mathbf{x})$ by an approximating function $F(\mathbf{W}, \mathbf{x})$ that has a fixed set of parameters $\mathbf{W}$. The approximation problem can be expressed as follows :

If $f(\mathbf{x})$ is a continuous function defined on $\mathbf{x}$, and $F(\mathbf{W}, \mathbf{x})$ is an approximating function that depends continuously on $\mathbf{W} \in P$ and $\mathbf{x}$, then the approximation problem is to define the parameters $\mathbf{W}^*$ such that

$$\rho[F(\mathbf{W}^*, \mathbf{x}), f(\mathbf{x})] \ \leq \ \rho[F(\mathbf{W}, \mathbf{x}), f(\mathbf{x})]$$

for all $\mathbf{W}$ in the set P. $\rho[\cdot, \cdot]$ is a distance function that evaluates a norm between two functions. $\mathbf{W}^*$ are the optimal parameter values of the approximating function.

When ρ is the L^2 norm, the above corresponds to minimizing the cost functional :

$$H[F(\mathbf{W}, \mathbf{x})] = \int_{-\infty}^{+\infty} (f(\mathbf{x}) - F(\mathbf{W}, \mathbf{x}))^2 d\mathbf{x}, \tag{1}$$

with respect to $\mathbf{W}$. In the above formulation of the approximation problem, the function $f(\mathbf{x})$ is continuous. However, in the case of learning a smooth mapping from a discrete set of examples there exists no continuous function $f(\mathbf{x})$. For this reason, the approximation problem is ill-posed for discrete data, the data does not contain sufficient information for a unique mapping.

The approximation problem is made well-posed by introducing apriori assumptions about the mapping. Normally the assumptions pertain to the smoothness of the mapping. Regularization techniques [12] are invoked to introduce smoothness constraints into the approximation problem. The resulting cost functional has the form :

$$H[F(\mathbf{W}, \mathbf{x})] = \sum_{i=1}^{N} (f(\mathbf{x}_i) - F(\mathbf{W}, \mathbf{x}_i))^2 + \lambda ||PF(\mathbf{W}, \mathbf{x})||^2, \tag{2}$$

where P is a differential operator, $\mathbf{x}_i$ are the N discrete points for which $f(\mathbf{x})$ is known, and λ is the regularization parameter that represents the tradeoff between enforcing the smoothness constraint and fitting the known data. Minimizing this functional using variational calculus leads to the associated Euler-Lagrange equations :

$$\hat{P}PF(\mathbf{W}, \mathbf{x}) = \frac{1}{\lambda} \sum_{i=1}^{N} (f(\mathbf{x}_i) - F(\mathbf{W}, \mathbf{x}_i)) \delta(\mathbf{x} - \mathbf{x}_i), \tag{3}$$

where $\hat{P}$ is the adjoint of the operator P.

The above is a partial differential equation whose solution can be written as the integral transform of the right side with a kernel given by the Green's function of $\hat{P}P$:

$$F(\mathbf{W}, \mathbf{x}) = \frac{1}{\lambda} \sum_{i=1}^{N} (f(\mathbf{x}_i) - F(\mathbf{W}, \mathbf{x}_i)) G(\mathbf{x}; \mathbf{x}_i), \tag{4}$$

where :

$$\hat{P}PG(x;y) = \delta(x-y).$$

The Green's functions, $G(\mathbf{x};\mathbf{x}_i)$, serve as the bases for the approximation scheme. When :

$$c_i = (f(\mathbf{x}_i) - F(\mathbf{W},\mathbf{x}_i))/\lambda,$$

the approximating function is :

$$F(\mathbf{W},\mathbf{x}) = \sum_{i=1}^{N} c_i G(\mathbf{x};\mathbf{x}_i). \tag{5}$$

The parameters, $\mathbf{W}$, include the coefficients c_i and the centers $\mathbf{x}_i$ of the Green's functions.

It is apparent from equation (3) the basis functions depend on the operator $\hat{P}P$. This operator is typically chosen to be both translationally and rotationally invariant. For this reason, the basis functions are rotationally and translationally invariant:

$$G(\mathbf{x};\mathbf{x}_i) = G(||\mathbf{x}-\mathbf{x}_i||).$$

These functions are radial basis functions and equation (5) can be rewritten as :

$$F(\mathbf{W},\mathbf{x}) = \sum_{i=1}^{N} c_i G(||\mathbf{x}-\mathbf{x}_i||). \tag{6}$$

The coefficients, c_i can be calculated by solving the following linear system:

$$\begin{pmatrix} f(\mathbf{x}_1) \\ \vdots \\ f(\mathbf{x}_N) \end{pmatrix} = \begin{pmatrix} G(\mathbf{x}_1;\mathbf{x}_1) & \cdots & G(\mathbf{x}_1;\mathbf{x}_N) \\ \vdots & \cdots & \vdots \\ G(\mathbf{x}_N;\mathbf{x}_1) & \cdots & G(\mathbf{x}_N;\mathbf{x}_N) \end{pmatrix} \begin{pmatrix} c_1 \\ \vdots \\ c_N \end{pmatrix}.$$

Let :

$$\mathbf{c} = \begin{pmatrix} c_1 \\ \vdots \\ c_N \end{pmatrix}, \mathbf{y} = \begin{pmatrix} f(\mathbf{x}_1) \\ \vdots \\ f(\mathbf{x}_N) \end{pmatrix}, \mathbf{G} = \begin{pmatrix} G(\mathbf{x}_1;\mathbf{x}_1) & \cdots & G(\mathbf{x}_1;\mathbf{x}_N) \\ \vdots & \cdots & \vdots \\ G(\mathbf{x}_N;\mathbf{x}_1) & \cdots & G(\mathbf{x}_N;\mathbf{x}_N) \end{pmatrix}.$$

There exists a solution for $\mathbf{c}$ as long as $\mathbf{G}$ is invertible :

$$\mathbf{c} = \mathbf{G}^{-1}\mathbf{y}. \tag{7}$$

This invertibility of $\mathbf{G}$ restricts the type of basis functions that can be used. If $\mathbf{G}$ is a positive definite matrix then it is invertible. Two theorems by Micchelli [10] exploit this property to impose sufficient conditions for the basis functions. The following are a few basis functions that satisfy Micchelli's conditions [12] :

$$\begin{aligned} G(r) &= e^{-r^2/\sigma^2} \\ G(r) &= \tfrac{1}{(c^2+r^2)^\alpha} \quad \alpha > 0 \\ G(r) &= (c^2+r^2)^\beta \quad 0 < \beta < 1 \\ G(r) &= r \end{aligned} \tag{8}$$

where, $r = ||\mathbf{x} - \mathbf{x}_i||$.

Once the type of basis function has been selected, the approximating function :

$$F(\mathbf{W}, \mathbf{x}) = \sum_{i=1}^{N} c_i G(\mathbf{x}; \mathbf{x}_i)$$

is easily cast as a network. The RBF [12] network has three layers, each fully connected to the next layer (see Figure 1). The first layer consists of a single input unit, the vector $\mathbf{x}$. The second layer is composed of the series of multidimensional radial basis functions $G(\mathbf{x}; \mathbf{x}_i)$. There exists one basis function for each data point $\mathbf{x}_i$. The third layer is the output, which is a weighted sum of the basis functions.

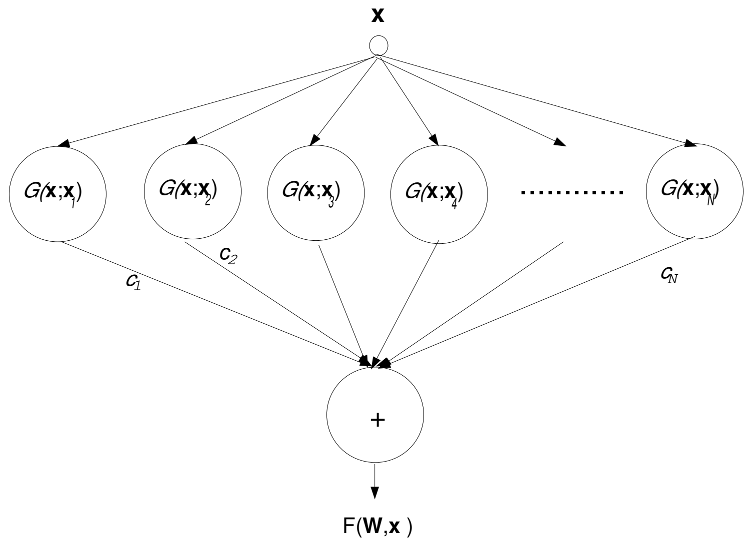

Figure 1: The RBF network has three layers. The first layer is the input vector, $\mathbf{x}$. The second layer consists of the radial basis functions $G(\mathbf{x}; \mathbf{x}_1)$ to $G(\mathbf{x}; \mathbf{x}_N)$. The c_i values are the weights between the i^{th} basis function and the output, $F(\mathbf{W}, \mathbf{x})$, which is the third layer.

The speed of the network in performing the input-output mapping depends upon the number of computations required. This is linear in the number of basis functions and the dimensions of the input and output space. In the above formulation the number of basis functions are set equal to the number of data points. Due to the large number of data points in most practical applications, the speed of the mapping becomes a serious limitation.

This problem can be avoided by using fewer basis functions than data points [12]. The approximating function, however, is no longer an exact representation of $f(\mathbf{x})$ and the approximation gets worse as the number of basis functions is reduced. The approximating function now appears as :

$$F(\mathbf{W}, \mathbf{x}) = \sum_{j=1}^{n} c_j G(\mathbf{x}; \mathbf{z}_j) \tag{9}$$

where $n < N$ (n is the number of basis functions and N is the number of data points), and $\mathbf{z}_j$ are the centers of the new basis functions (the centers of the basis functions no longer have to be at data points). The coefficients, c_i, are calculated in a similar fashion as in equation (7). However, the pseudo-inverse is used instead:

$$\mathbf{c} = (\mathbf{G}^T\mathbf{G})^{-1}\mathbf{G}^T\mathbf{y}. \tag{10}$$

This is equivalent to performing a least-means-square fit of $F(\mathbf{W}, \mathbf{x})$ to $f(\mathbf{x})$. In addition we can vary the positions of the basis functions to minimize the cost functional :

$$H[F(\mathbf{W}, \mathbf{x})] = \sum_{i=1}^{N}(F(\mathbf{W}, \mathbf{x}_j) - f(\mathbf{x}_j))^2. \tag{11}$$

This is done by setting :

$$\frac{\partial H[F(\mathbf{W}, \mathbf{x})]}{\partial \mathbf{z}_j} = 0,$$

where $j = 1...n$. This type of network is called a Generalized Radial Basis Function Network (GRBF). The parameters of these networks have traditionally been set in an ad hoc manner. A programmer arbitrarily choses the number n of basis functions and provides initial estimates for the positions z_j and the spans of these bases. Then, conjugate gradient or gradient descent algorithms are used to optimize the position and span parameters. Next, the pseudoinverse is then used to calculate the coefficients c_j of the bases. If the approximation is not within the given error bound the number of basis functions is increased and the procedure is repeated. The above approach greatly sacrifices the theoretical structure that makes RBF networks appealing. Furthermore, the use of optimization methods makes this process cumbersome and time consuming. In the next section we introduce a technique for determining network parameters that avoids these shortcomings.

3 Integral Wavelet Transforms and RBF Networks

In this section, the use of an integral wavelet transform [5] to set the parameters of a RBF network is developed. It shall be shown that by using the transform, the number of basis functions required for a given mapping and error bound can be determined analytically. In addition, the spans, positions, and coefficients of the bases are directly computed by the transform. Eventually, these parameters are used to construct a RBF network.

3.1 The Wavelet Transform

The integral wavelet transform (IWT) is a generalization of the principle underlying the Fourier transform. In the case of the Fourier transform, a function is decomposed into a series of weighted sines and cosines of different frequencies. The sines and cosines therefore serve as the basis functions. The most important property of these bases is that they are orthonormal. Using the IWT we can construct orthonormal basis functions that are localized in space and then decompose a function in terms of these bases. For the case of 1-dimensional functions, $\mathcal{R}^1 \mapsto \mathcal{R}^1$. The IWT has the basic form (see [5]) :

$$(T_\psi f)(b, a) = \int_{-\infty}^{+\infty} \psi_{b,a}(x) f(x) dx$$

where

$$\psi_{b,a}(x) = |a|^{-1/2}\psi(\frac{x-b}{a})$$

are the bases called wavelets. The function $f(x)$ is characterized by the following: (a) The location of a change in $f(x)$ in terms of b; (b) The rate in the change in $f(x)$ in terms of a; (c) The amount of this change in terms of $(T_\psi f)(b,a)$. In addition, the function $f(x)$ can be reconstructed from the basis functions :

$$f(x) = \sum_a \sum_b (T_\psi f)(b,a)\, |a|^{-1/2}\, \psi(\frac{x-b}{a}).$$

The IWT allows us to decompose functions at different resolution levels, from fine to coarse. The accuracy in reconstructing the original function decreases as one goes to coarser resolution levels. The basis functions at all resolution levels are either orthonormal or biorthonormal to each other. Two functions are involved in a wavelet transform, a scaling function and a wavelet. The a and b parameters of the wavelet are discretized:

$$a = \frac{1}{2^j} \text{ and } b = \frac{k}{2^j},\ j \in Z$$

where j is the scale parameter and k is the position parameter. The scaling function is written as

$$\phi_{j,k}(x) = 2^{j/2}\phi(2^{-j}x - k).$$

These scaling functions are then used to construct the wavelet bases and scaling functions at a coarser resolution level:

$$\psi_{j-1,k} = \sum_k q(k)\phi_{j,k} \tag{12}$$

$$\phi_{j-1,k} = \sum_k p(k)\phi_{j,k}. \tag{13}$$

The $q(k)$ values are chosen such that

$$< \psi_{j,k}(x), \phi_{j,k}(x) >= 0.$$

Because of the above construction, the wavelet bases are orthonormal across scale. If the $q(k)$ values are selected carefully, the wavelet bases are orthonormal across position. When this is not the case, the bases can be made biorthonormal. The result is a set of orthonormal bases :

$$< \psi_{j,k}, \psi_{l,m} >= \delta_{j,l}\, \delta_{k,m}.$$

An approximation to a function $f(x)$ exists at different resolution levels. At the resolution level, j, the approximating function has the form [5]:

$$f_j(x) = \sum_k c_j(k)\, \phi(2^{-j}x - k) + \sum_k d_j(k)\, \psi(2^{-j}x - k). \tag{14}$$

From the above, it can be shown that the exact representation of the function $f(x)$ has the form [5]:

$$f(x) = \sum_j \sum_k d_j(k)\psi(2^{-j}x - k). \tag{15}$$

where :

$$d_j(k) = < f(k), \psi(2^{-j}x - k) > . \tag{16}$$

The decomposition in equation (16) is equivalent to a weighted sum of scaling functions at the finest resolution level:

$$f(x) = \sum_{k=1}^{N} c(k)\phi_0(x - k). \tag{17}$$

The wavelet transform allows us to create a variety of localized orthonormal bases depending on the scaling function used. If we use a scaling function that approximates a radial basis function then equation (17) is identical to the approximating function of equation (5) in section 2. One group of scaling functions that approximate radial basis functions are β-splines of order greater than one (see Appendix A for details). Our approach is to apply the IWT to the training data of the input-output mapping problem and to use the transform coefficients and the wavelet bases to construct a RBF network.

3.2 Relevant Properties of the Transform

The training data given, $x_i \mapsto f(x_i)$, can be considered a discrete signal. So a Discrete Integral Wavelet Transform [9] is implemented. The input space is discretized into 2^J bins, where J typically ranges from 9 to 11. The $f(x_i)$ values are then placed into the appropriate bin by rounding off x_i. The result is a signal of the form $f(k)$ where $k \in Z$. Since the bins are small, we assume that any error introduced from the rounding off is negligible.

The next step involves calculating the transform coefficients, $d_{j,k}$, for the wavelet bases, $\psi_{j,k}$. A variation of Mallat's fast wavelet algorithm [9] is used to calculate these coefficients. We extend Mallat's algorithm to the case of unevenly sampled discrete signals. This algorithm is pyramidal in nature [9] :

$$c_{j-1}(k) = [\mathring{v} * c_j]_{\downarrow 2}(k) \tag{18}$$

$$d_{j-1}(k) = [\mathring{w} * c_j]_{\downarrow 2}(k) \tag{19}$$

$\downarrow 2$ denotes downsampling by two, keep every other term. The c_0 values correspond to the signal at the finest resolution level, the discrete function $f(k)$. The formulas for $\mathring{v}$ and $\mathring{w}$ for the Battle-Lamarié basis [9] constructed from cubic β-splines are given in Appendix B. This algorithm is iterated for $j = 0$ to $1 - J$, where $j = 0$ is the finest resolution level and $j = 1 - J$ is the coarsest level.

Since little is known about the training data, we assume that $f(k)$ need not be evenly sampled. This brings up the question of how to implement:

$$a(x) = f(x) * g(x) \tag{20}$$

when $f(x)$ is an unevenly sampled discrete signal and $g(x)$ is a continuous function. A solution to this problem requires an extension to Mallat's fast wavelet algorithm [9].

Spectral analysis of unevenly sampled data has been explored by astronomers. The most frequently used method is the Lomb periodogram [8]. We use it to help define the operation in equation (20). The periodogram performs a least-mean-square fit of the data to sines and cosines under the assumption that the error in the fit decreases with the addition of frequency terms. In our implementation, the data was sampled at N_p frequencies :

$$N_p = \frac{f_h}{2} N,$$

$f_h = \frac{f_{hi}}{f_c}$ where $f_c = \frac{1}{2\Delta x_{avg}}$, $f_{hi} = \frac{1}{2\Delta x_{min}}$, Δx_{min} is the smallest distance between two data points, and Δx_{avg} is the average distance between data points. The decomposition of $f(x)$ by the periodogram has the form :

$$f(x) = \sum_{m=0}^{N_p} q_m e^{i\omega_o m x} \tag{21}$$

where

$$\begin{aligned} q_m &= < f(x), e^{-i\omega_o m x} > \\ \omega_o &= \frac{\pi}{T}. \end{aligned} \tag{22}$$

T is the total length of the discrete data. The Lomb periodogram reduces to the Discrete Fourier transform when $f(x)$ is evenly sampled.

Using the periodogram we can map $f(x) \mapsto F(\omega)$, where $F(\omega)$ are the values q_m determined by the periodogram. An assumption is then made that there exists a unique continuous function $h(x)$ which has the same spectral properties as $f(x)$, i.e. $H(\omega) = F(\omega)$. The two functions $f(x)$ and $h(x)$ have identical values at all x_i for which $f(x_i)$ is given. This equivalence of the functions in both spatial and frequency domains is a result of the oversampling of $f(x)$ in the periodogram. The continuous function $h(x)$ can be derived :

$$\begin{aligned} h(x) &= \frac{1}{2\pi} \int_{-\infty}^{+\infty} H(\omega) e^{i\omega x}\, d\omega \\ &= \frac{1}{2\pi} \int_{-\infty}^{+\infty} F(\omega) e^{i\omega x}\, d\omega \\ &= \frac{1}{2\pi} \sum_{m=0}^{N_p} F(\omega_m) e^{i\omega_o m x}. \end{aligned} \tag{23}$$

The continuous function $h(x)$ replaces $f(x)$ in the convolution in equation (20) so that the operation $a(x) = h(x) * g(x)$ can be performed.

In the fast wavelet algorithm [9], the above technique can be used to perform the convolutions in equations (18) and (19). The unevenly sampled $c_j(k)$ is replaced by the continuous $c'_j(x)$ and $C_j(\omega) = C'_j(\omega)$. Since

$$\overset{\circ}{v}(k) * c'_j(k) = \overset{\circ}{V}(\omega) C'_j(\omega)$$

we can calculate a continuous function $c'_{j-1}(x)$:

$$\begin{aligned} C'_{j-1}(\omega) &= \sum_{m=0}^{N_p} \overset{\circ}{V}(\omega_m) C'_j(\omega_m) \\ c'_{j-1}(x) &= \frac{1}{2\pi} \sum_{m=0}^{N_p} \overset{\circ}{V}(\omega_m) C'_j(\omega_m)\, e^{i\omega_o m x}. \end{aligned} \tag{24}$$

This continuous function $c'_{j-1}(x)$ is sampled at the resolution level 2^j and then downsampled by 2 to obtain $c_{j-1}(k)$:

$$c_{j-1}(k) = [c'_{j-1}(l/2)]_{\downarrow 2} \tag{25}$$

where $l = 1$ to m (where m is the maximum index minus the minimum index at which there exists a known signal value). The same procedure is used to calculate $d_{j-1}(k)$.

This variation of the fast wavelet algorithm allows the decomposition of unevenly sampled data. The weakness in this approach is the assumption inherent in the periodogram that the sampling rate exceeds what would be the Nyquist frequency if the data were evenly sampled. A discussion of what approximations are introduced due to this oversampling assumption is beyond the scope of this paper. A rigorous approach to performing wavelet transforms on non-uniform data has been developed by Buhmann and Micchelli [3] [4]. We have not used their approach for two reasons. First, extending their approach to multidimensional problems results in a tremendous number of computations. Secondly, they introduce an extra spline space of radial basis functions called prewavelets to perform the transform. This extra space makes it difficult to construct a network once the parameters are computed.

The mappings considered so far in this section have been $\mathcal{R}^1 \mapsto \mathcal{R}^1$. To extend the transform to $\mathcal{R}^n \mapsto \mathcal{R}^m$, multidimensional functions, $\Phi(\mathbf{x})$ and $\Psi(\mathbf{x})$, are introduced for both the scaling and wavelet functions :

$$\begin{aligned} \Phi(\mathbf{x}) &= \phi(x_1)\phi(x_2)....\phi(x_n) \\ \Psi(\mathbf{x}) &= \psi(x_1)\psi(x_2)....\psi(x_n). \end{aligned}$$

Both $\Psi(\mathbf{x})$ and $\Phi(\mathbf{x})$ are tensor product splines since they are separable across each dimension x_d. The decomposition of $f(\mathbf{x})$ becomes

$$f(\mathbf{x}) = \sum_j \sum_{k_1(j)} ... \sum_{k_n(j)} d_{j,\mathbf{k}}(\mathbf{x}) \psi(2^{-j}x_1 - k_1)....\psi(2^{-j}x_n - k_n) \tag{26}$$

where $\mathbf{k}$ corresponds to the vector $[k_1 k_2 ... k_n]$. Since $\Psi(\mathbf{x})$ and $\Phi(\mathbf{x})$ are separable, the coefficients $d_{j,\mathbf{k}}$ can be calculated by applying the fast wavelet algorithm across each dimension. The function $\mathbf{f}(\mathbf{x})$ is equivalent to the vector $[f_1(\mathbf{x}), ..., f_l(\mathbf{x}), ...f_m(\mathbf{x})]$ where each $f_i(\mathbf{x})$ can be considered a separate signal. The transform is performed on each signal. The decomposition of a function $\mathbf{f}(\mathbf{x})$ has the form:

$$f_1(\mathbf{x}) = \sum_j \sum_{k_1(j)} ... \sum_{k_n(j)} d_{j,\mathbf{k},1}(\mathbf{x}) \psi(2^{-j}x_1 - k_1)....\psi(2^{-j}x_n - k_n)$$

$$f_2(\mathbf{x}) = \sum_j \sum_{k_1(j)} ... \sum_{k_n(j)} d_{j,\mathbf{k},2}(\mathbf{x})\psi(2^{-j}x_1 - k_1)....\psi(2^{-j}x_n - k_n) \tag{27}$$

$$\vdots$$

$$f_m(\mathbf{x}) = \sum_j \sum_{k_1(j)} ... \sum_{k_n(j)} d_{j,\mathbf{k},m}(\mathbf{x})\psi(2^{-j}x_1 - k_1)....\psi(2^{-j}x_n - k_n)$$

where $d_{j,\mathbf{k},l}$ is the transform coefficient for the l^{th} output at the $2^{j\,th}$ resolution level for the $\mathbf{k}^{th}$ basis. The result of the transform applied to a mapping from $\mathcal{R}^n \mapsto \mathcal{R}^m$ are the coefficients $d_{j,\mathbf{k},l}$. The magnitudes of these coefficients tells us how important the wavelet corresponding to the coefficient is in approximating the function.

3.3 Using the Transform to Construct a RBF Network

The result of applying the transform to a mapping from $\mathcal{R}^n \mapsto \mathcal{R}^m$ are the coefficients $d_{j,\mathbf{k},l}$. Using these coefficients and the error bound specified, a RBF network is constructed. Assume that the L^2 norm between a function $\mathbf{f}(\mathbf{x})$ and the approximating function $\mathbf{F}(\mathbf{W},\mathbf{x})$ must be less than the error bound, ϵ :

$$\epsilon > \sum_{i=1}^{N}(\mathbf{f}(\mathbf{x}_i) - \mathbf{F}(\mathbf{W},\mathbf{x}_i))^2 \tag{28}$$

where N is the number of known values of $\mathbf{f}(\mathbf{x})$. Since the wavelet bases are orthonormal, Parseval's theorem can be used :

$$P_t = \sum_{i=0}^{N} ||\mathbf{f}(\mathbf{x}_i)||^2 = \sum_{l=1}^{m}\sum_{j=0}^{J-1}\sum_{k_1=0}^{K_1(j)} ... \sum_{k_n=0}^{K_n(j)} d^2_{j,\mathbf{k},l}$$

where m is the dimensionality of the output, $J-1$ is the coarsest resolution level, $K_d(j)$ are the number of bases at the resolution level j in the dimension x_d, and P_t is the total energy in the signal. The energy in the approximating function $\mathbf{F}(\mathbf{W},\mathbf{x})$ is :

$$||\mathbf{F}(\mathbf{W},\mathbf{x})||^2 = \sum_{l=1}^{m}\sum_{j=0}^{J-1}\sum_{k_1=0}^{K'_1(j)} ... \sum_{k_n=0}^{K'_n(j)} d^2_{j,\mathbf{k},l} \tag{29}$$

where $K'_d(j) < K_d(j)$. This is equivalent to setting the coefficients for some of the wavelet bases to 0. Using equation (29), equation (28) can be rewritten as :

$$\epsilon^2 > 1 - \frac{1}{P_t}\sum_{l=1}^{m}\sum_{j=0}^{J-1}\sum_{k_1=0}^{K'_1(j)} ... \sum_{k_n=0}^{K'_n(j)} d^2_{j,\mathbf{k},l}. \tag{30}$$

The number of wavelet bases used in the approximation for a given ϵ is calculated by finding $K'_d(j)$ at each resolution level j and each dimension x_d until the above condition is met. One way of doing this is by calculating sums of the coefficients over the output dimensions:

$$s_{j,\mathbf{k}} = \sum_{l=1}^{m} d^2_{j,\mathbf{k},l}$$

these sums $s_{j,\mathbf{k}}$ are then ordered from 1 to M so that $s_{j(1),\mathbf{k}(1)} \geq s_{j(2),\mathbf{k}(2)} \geq ... s_{j(M),\mathbf{k}(M)}$ where M is the number of wavelet bases in the decomposition of $\mathbf{f}(\mathbf{x})$. Equation (30) can now be expressed as follows:

$$\epsilon^2 > 1 - \frac{1}{P_t} \sum_{i=1}^{M'} s_{j(i),\mathbf{k}(i)} \tag{31}$$

where M' is the smallest integer that satisfies the above condition. It is calculated by adding the $s_{j(i),\mathbf{k}(i)}$ terms until the inequality in equation (31) is satisfied. The resulting approximating function is:

$$\mathbf{F}(\mathbf{W}, \mathbf{x}) = \sum_{i=1}^{M'} \mathbf{g}_{j(i),\mathbf{k}(i)} \, \Psi_{j(i),\mathbf{k}(i)}(\mathbf{x}) \tag{32}$$

where:

$$\mathbf{g}_{j(i),\mathbf{k}(i)} = \begin{pmatrix} d_{j(i),\mathbf{k}(i),1} \\ \vdots \\ d_{j(i),\mathbf{k}(i),m} \end{pmatrix}.$$

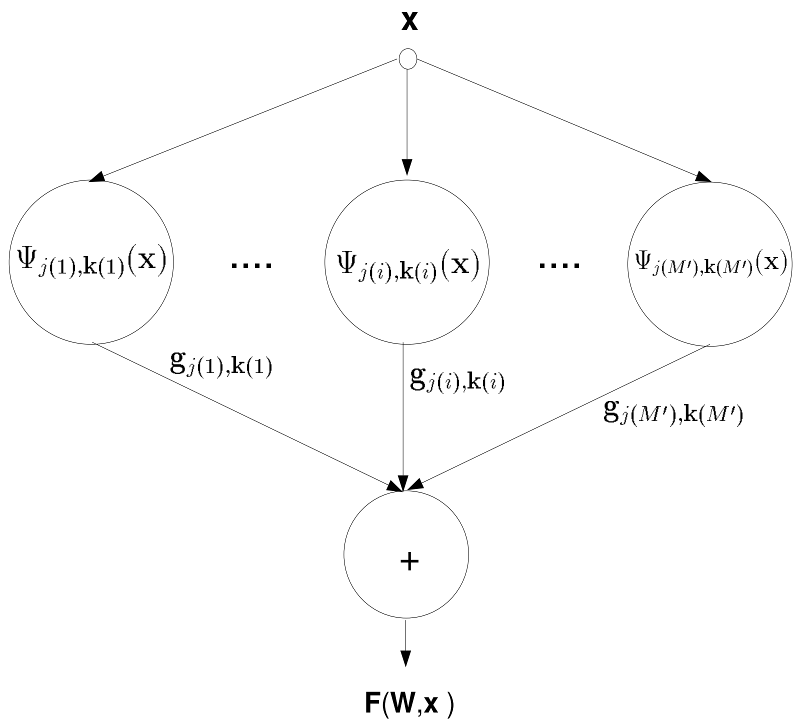

Figure 2: The RBF network derived using the wavelet transform. The first layer is the input vector, $\mathbf{x}$. The second layer consists of the wavelet bases $\Psi_{j(1),\mathbf{k}(1)}$ to $\Psi_{j(M'),\mathbf{k}(M')}$. The $\mathbf{g}_{j(i),\mathbf{k}(i)}$ are the weights between the i^{th} wavelet basis and the output, $\mathbf{F}(\mathbf{W}, \mathbf{x})$.

The expression in equation (32) can be easily be mapped to a network as shown in Figure 2. The wavelet bases, $\Psi_{j(i),\mathbf{k}(i)}$, are not radial basis functions. However, each wavelet basis is a weighted sum of scaling functions at a finer level. For this reason, we can consider this network to be a RBF network.

3.4 Summary

Using the integral wavelet transform to determine the parameters of a RBF network is a powerful tool. With this approach, the number of basis functions required to satisfy a given error bound are determined analytically. In addition, the parameters of the network are also determined by the transform. This results in a more efficient network with respect to time and memory. The rigor of RBF networks is not sacrificed since the ad hoc and cumbersome aspects of conventional RBF network implementations are avoided.

4 Computational Complexity and Memory Requirements

The two main factors that determine the usefulness of a RBF network is the time required to perform the mapping and the memory required to store the mapping. The time required for the network to learn the mapping is not as vital an issue since it is done off-line. The above factors are related to the dimensionality of the input and output spaces and the error bound.

The speed of the mapping depends on the number of computations which, in turn, is determined by the dimensionality of the input and output spaces and the number of basis functions in the approximating function. The number of computations required to evaluate the output of the wavelet basis also effects the speed of the mapping. For this reason, the basis functions are stored as a look-up table, avoiding the need for numerous calculations. The computational complexity of the proposed RBF network can be expressed as follows :

$$C_{net} = 6 n_o n_i N \tag{33}$$

where C_{net} is the number of additions and multiplications performed by the network, n_i is the dimensionality of the input space, n_o is the dimensionality of the output space, and N is the number of basis functions in the network. The complexity is $O(mN)$ where m is the dimensionality of the mapping, $m = n_i \times n_o$.

The second issue is the amount of memory required to store the mapping. This depends again on the dimensionality of the input and output spaces, the number of basis functions, and the resolution of the look-up table used for the basis functions. All the basis functions in the network are translations and dilations of an orthonormal wavelet. This allows us to use a single look-up table and two sets of normalization factors (one set for the translation and one set for scaling) for each basis. For each basis function, $n_i + 1$ integers (n_i integers for the position of the basis in each input dimension and an extra integer for the scaling factor) and n_o doubles (the weights for the basis function) are stored. In addition, the look-up table needs to be stored. The memory required is determined by the resolution of look-up table, n_{res}. For a Sparc IPX (where a double is 8 bytes and an integer is 4 bytes) the amount of memory, M_{net}, required to store network parameters is :

$$M_{net} = 4 \times N \times (n_i + 1) + 8 \times N \times n_o + 8 \times n_{res} \ \ bytes. \tag{34}$$

5 A Simple Example

A simple 1-dimensional example is presented to demonstrate the advantage of using the transform technique over traditional optimization methods. These simulations are performed on two discretized 1-dimensional functions. The first function is $f_1(x) = sin(2\pi n/64)$ where n ranges from 0 to 63. Note that this function is smooth and infinitely differentiable. The second function, a bifurcating series, is $x_{n+1} = 3x_n(1 - x_n)$ where n ranges from 0 to 63 and $x_0 = .4467$. This function is not smooth and converges to an oscillation between two values. The RBF networks constructed for these two functions using the transform approach will be referred to as *optimal networks* henceforth. For a comparison, RBF networks are also constructed using a traditional algorithm, referred to as *conventional networks*. The specific algorithm used to construct the conventional networks is outlined below :

step (1) Initialize the number of basis functions, $n = 1$.

step (2) Set the values for the centers of the n basis functions to an arbitrary subset of the N data points.

step (3) Set the spans of the basis functions to $\frac{\Delta x}{n}$ where Δx is the span of the input space

step. (4) Calculate the coefficients, c_i, using the pseudo-inverse (equation 10).

step (5) Use conjugate gradient decent to adjust the center values.

step (6) Return to step 4 until the change in the error functional between two iterations is negligible.

step (7) If the error functional is greater than the specified error bound, increment n and return to step (2).

In the above algorithm, the spans of the basis functions were equal and not allowed to vary as it took too much time to search the parameter spaces corresponding to both position and span. The above algorithm is commonly used to set RBF network parameters [12]. The accuracy of the optimal and conventional networks as a function of the number of bases, for both functions, is shown in Figure (3). For both functions, the error functional decays faster for the optimal network. In the case of the sine function the improvement is negligible. The performance of the optimal network is dramatically superior for the bifurcating series. The error functional does not decrease monotonically for the conventional network. This is because the conjugate gradient decent method is susceptible to the problem of local minima in the parameter space. Therefore, incrementing the number of basis functions does not necessarily decrease the error functional since the parameters are prone to be trapped in local minima.

The parameters of the optimal network are calculated much quicker than those of the conventional network. For the example functions above, the parameters for the optimal network are determined within 9.06sec, independent of the number of basis functions. In contrast, it can take between 10.04sec and 20min to set the parameters of the conventional network, depending on the number of basis functions used. The conjugate gradient decent method is of order $O(n^2)$ where n is the number of basis functions. The wavelet transform is of order $O(d\log_2(d))$, where d is the number of known samples of the function. Therefore, the optimal network is constructed not only more accurately but also faster than the conventional network. From the point of view of recognition the optimal network takes less memory and is faster than the conventional network since it requires fewer basis functions for any given accuracy. Both networks have the same computational complexity for recognition. Figure (4) shows the reconstruction of the two example functions using the two types of networks. It is apparent that the performance of the optimal network is superior to that of the conventional network in terms of the error functional for the bifurcating series. For the sine function the optimal network is slightly more accurate.

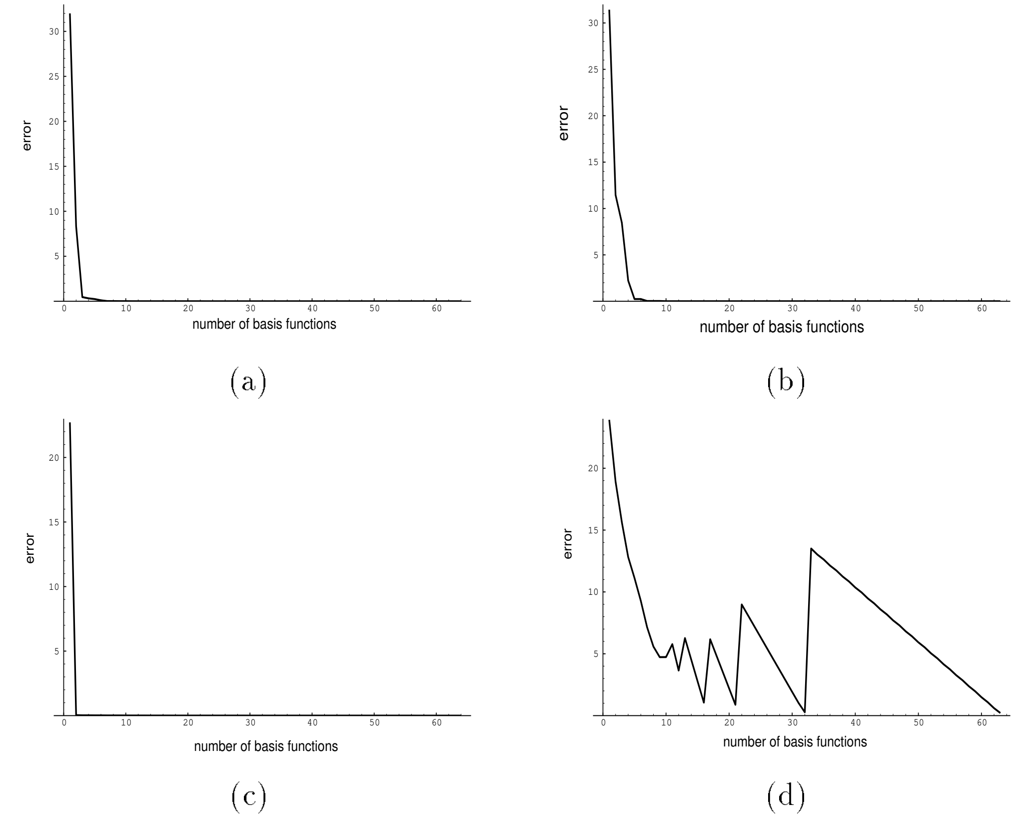

Figure 3: The decay in L^2 error in the approximation of $f(x) = sin(2\pi x/64)$ as the number of basis functions increases for (a) the optimal network and (b) the conventional network. The decay in L^2 error in the approximation of $x_{i+1} = 3x_i(1 - x_i)$ where $x_o = .4467$ as the number of basis functions increases for (c) the optimal network and (d) the conventional network.

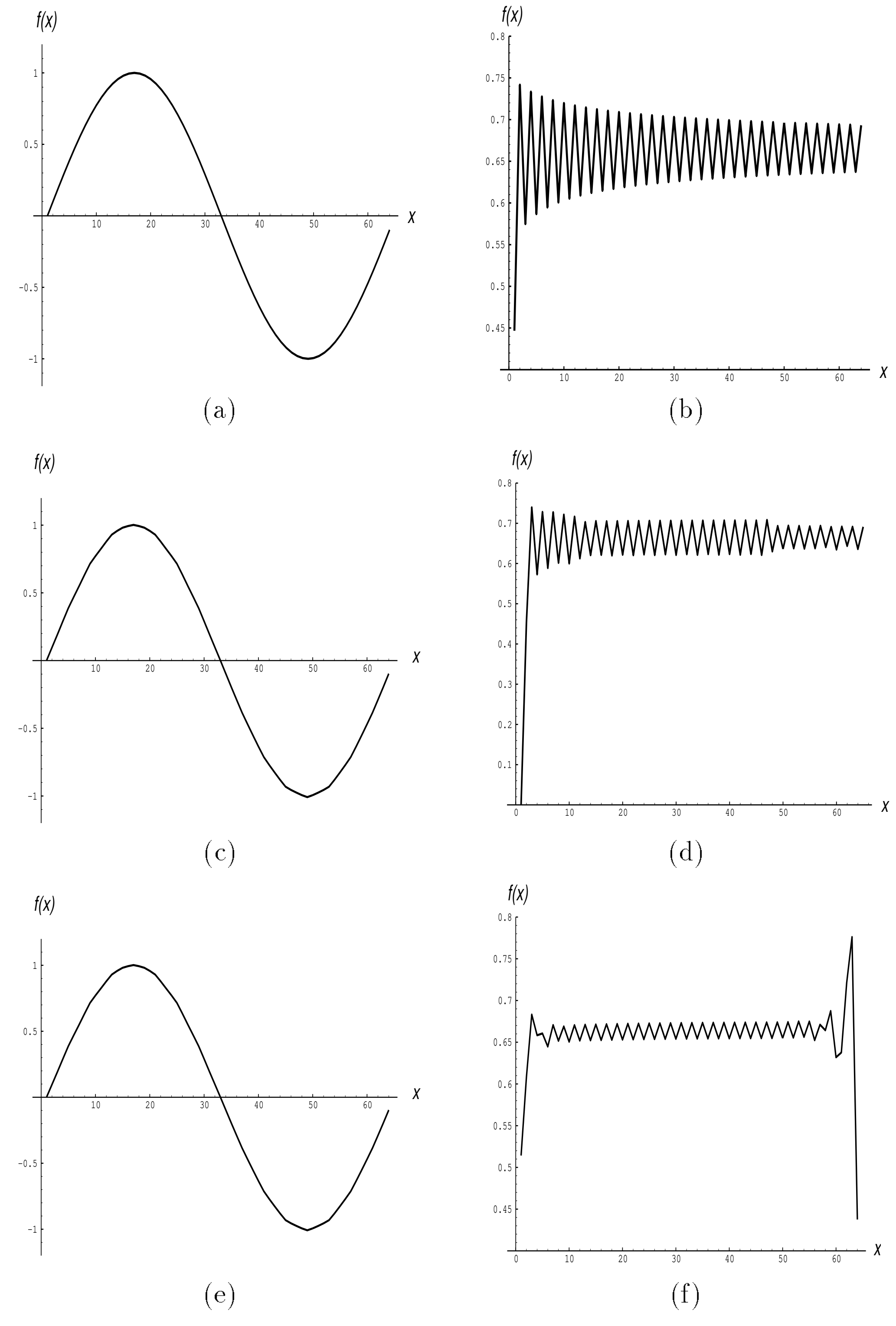

Figure 4: (a) The function $f(x) = sin(2\pi x/64)$. (b) The series $x_{i+1} = 3x_i(1 - x_i)$ where $x_0 = 0.4467$. (c) Reconstruction of the sine function by an optimal network with 8 basis functions. (d) Reconstruction of the series with an optimal network with 32 basis functions. (e) Reconstruction of the sine function with a conventional network with 8 basis functions. (f) Reconstruction of the series with a conventional network with 32 basis functions.

6 Experiments

The following experiments were conducted to test the performance of optimal networks for multidimensional mappings. The experiments involve a network that recognizes an object and estimates its pose in a scene. Murase and Nayar [11] developed a system that uses principal component analysis to a create a compact representation of object appearance parameterized by pose. Since our network is used to enhance the performance of the methodology proposed by Murase and Nayar, a brief overview of their work is in order. During the learning stage, for each object, a large image set is acquired by varying pose. The eigenvectors of the correlation matrix of this image set, corresponding to the largest eigenvalues, make up the dimensions of a subspace (typically 10-20 dimensions) called the eigenspace. When such a subspace is computed using image sets of all objects, it is referred to as the universal eigenspace. Each image of an object is projected to the universal eigenspace, by taking the dot product of the image with the eigenvectors. This results in a single point in eigenspace. The projections of all images of an object results in a set of points (corresponding to the different discrete poses), that is referred to as a *discrete manifold.* In [11], the discrete manifold is interpolated using biquadratic splines to obtain a continuous manifold that is parameterized by object pose. The manifold is densely resampled to get a large number of manifold points. This large point set represents that object's appearance model. The above process is repeated for all objects of interest to the recognition system.

Given a novel object image, the object region is segmented, normalized in scale and projected to universal eigenspace. The closest manifold determines the identity of the object in the image and the exact position of the new projection on the manifold yields the pose of the object. In [11], the closest manifold point is determined either by exhaustive search (which is inefficient in time and memory) or by binary search (which is inefficient in memory).

Our objective is not to interpolate and resample discrete manifolds of the objects, but rather to develop a RBF network (for each object as shown in Figure 5) that performs a continuous mapping from a discrete manifold point to a confidence value C_p that represents the likelihood that the novel image is that of the object for which the network has been developed, and the pose θ_p of the object. This network is generated automatically from the discrete manifold of an object using the transform technique developed in this paper. The result is a set of P optimal networks where P is the total number of objects. During recognition the input, an eigenspace projection, is mapped by each of the networks and the network that produces the highest confidence value reveals the identity of the object and its pose.

Note that the pose of each object has a discontinuity at $\theta_p = 360^o$. The size of a RBF network is clearly related to the continuity of the function it seeks to approximate (this topic will be revisited in section 7). Therefore, judicious selection of data representation can dramatically reduce the size of the network. In the present application, we benefited by using $\sin(\theta_p)$ and $\cos(\theta_p)$ as outputs, instead of using θ_p. This eliminates the ill-conditioning that results from the discontinuity in θ_p. During recognition, the network with the highest confidence value C_p is first identified and if this value exceed a threshold level (i.e. if the projection is close enough to the object's manifold), then pose is computed from $\sin(\theta_p)$ and

$\cos(\theta_p)$ using arctan.

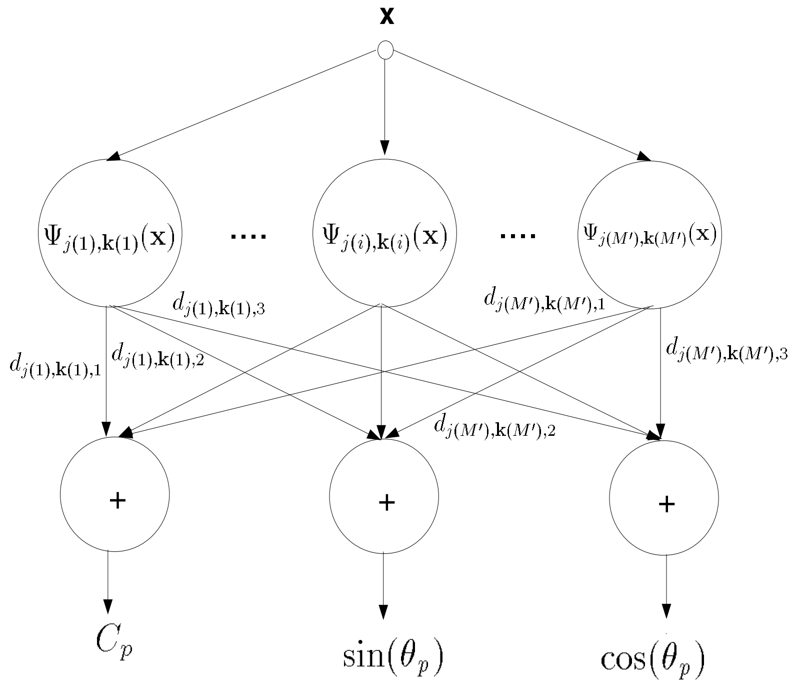

Figure 5: An optimal network for a particular object, p. The input is a point in eigenspace and the outputs are C_p, $\sin(\theta_p)$, and $\cos(\theta_p)$.

In our experiments, an optimal network was constructed for each of the 20 objects used by Nayar and Murase [11] (see Figure 6). The input space, a 15-dimensional eigenspace, was discretized into 1024 boxes in each of its dimensions. The networks ability to learn and generalize examples presented to it was tested using two data sets. The *training set* includes 36 discrete manifold points (poses) for each object and was used to generate the networks. The *test data* includes a different set of 36 manifold points and was used to test the accuracy of the networks and their ability to generalize to data not seen before.

The most important task of the set of networks is to correctly recognize objects in the test set. In our experiments, every point in both the training and test sets were correctly recognized yielding a 100% recognition rate. We found that for any object v, $0.842 \leq C_p \leq 1.217$ when $v = p$ and $0.0 \leq C_p \leq 0.211$ when $v \neq p$, leading to robust object identification.

The networks' accuracy in pose estimation was also studied. For each object ,three networks (network$_1$, network$_2$, and network$_3$) were constructed using the data in the training set, each with a different error bound. The error bound for the three networks was set such that average absolute pose error for network$_1$ was 0.5^o, 1.0^o for network$_2$, and 1.5^o for network$_3$, for each object. The accuracy of pose estimation is defined as the absolute difference between the known pose and the pose estimated by the network. It is computed for both the test set and the training set and the results are summarized in Figure 7. As expected, the average errors in pose estimation are greater for the test set than the training set. A comparison of the performance of the above three networks is summarized in Table 1.

In multidimensional learning problems, it is standard practice to examine the effect of the dimensionality of the input space on the accuracy of the mapping. Four optimal networks corresponding to object number 10 (the jar of Vaseline) were constructed from its training set. The input spaces for these networks were 5, 10, 15 and 20 dimensional, respectively. As before, the output of each network is 3-dimensional, including the parameters C_p, $\sin(\theta_p)$, and $\cos(\theta_p)$. For the error bounds specified, each network ended up with 11 wavelet basis functions. The accuracy of all four networks in addition to time required for learning and recognition are summarized in Table 2. As expected, the average error in pose estimation increases as the dimensionality of the input space decreases. The recognition time is linear with respect to the dimensionality of the input space, as predicted by the complexity analysis in section 4. The time requirement for learning network seems to increase as a polynomial function of the input dimensionality, however further investigation is needed to state anything more definitive.

Figure 6: The twenty objects in the object recognition and pose estimation system [11]. The image set for each object includes 72 images corresponding to uniformly sampled discrete poses between 0^o and 360^o. Half of these points were used as training data to generate RBF networks and the other half for testing the networks.

	time	average accuracy over test set	average accuracy over training set	memory
$network_1$	211msec	0.58^o	0.31^o	40 kbytes
$network_2$	89msec	1.13^o	0.85^o	23 kbytes
$network_3$	58msec	1.63^o	1.36^o	13 kbytes

Table 1: The performance of optimal networks for all objects in terms of the time, accuracy, and memory involved in recognition is summarized above.

	learning time	recognition time	average accuracy over test set	average accuracy over training set
$\mathcal{R}^5 \mapsto \mathcal{R}^3$	72sec	$\sim$ 1msec	5.37^o	4.21^o
$\mathcal{R}^{10} \mapsto \mathcal{R}^3$	145sec	$\sim$ 2msec	4.21^o	3.11^o
$\mathcal{R}^{15} \mapsto \mathcal{R}^3$	214sec	$\sim$ 3msec	1.92^o	1.66^o
$\mathcal{R}^{20} \mapsto \mathcal{R}^3$	289sec	$\sim$ 4msec	1.43^o	1.32^o

Table 2: The effect of the dimensionality of the input space on the accuracy, recognition time, and learning time for optimal networks.

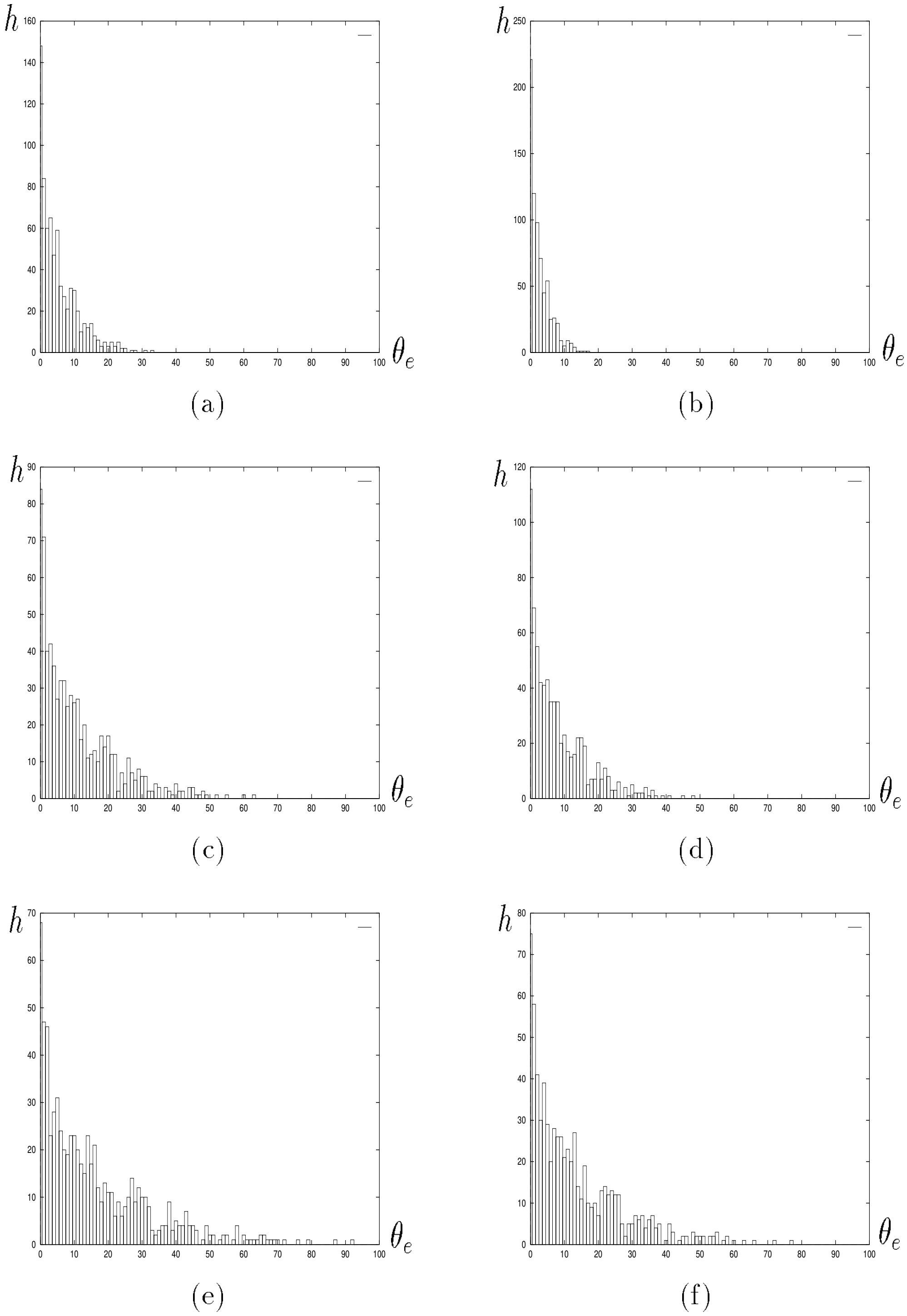

Figure 7: Histogram of the absolute value of the error, θ_e, between poses estimated by network$_1$ and the known pose, for (a) the test set and (b) the training set. Pose estimation histogram for network$_2$ for (c) the test set and (d) the training set. Pose estimation histogram for network$_3$ for (e) the test set and (f) the training set. Each bar in the histogram corresponds to 0.1^o.

7 Discussion

We have introduced a novel approach to setting RBF network parameters using the integral wavelet transform. This approach allows the automatic generation of a RBF network given an error bound. In addition, the network generated is the smallest network that performs the mapping for the error bound. The advantage of the transform approach over conventional optimization based techniques is demonstrated using two 1-dimensional functions as well as the multidimensional problem of 3-dimensional object recognition and pose estimation. The networks generated by the transform approach outperformed the other networks. The difference in performance was often dramatic.

The use of RBF networks is not limited to high-level vision processes like recognition. This type of network can be used to predict chaotic time series, recognize and synthesize speech signals, and control robot manipulators. Any supervised learning problem that can be formulated from variational and regularization principles is well suited for the automatic network generation introduced in this paper. Therefore, the ideas presented here have far reaching implications. Some of the issues that arose in developing this method that need further investigation discussed below :

- **Continuity Conditions of the Basis and the Mapping** : If the continuity conditions of a mapping are known there should be a way to find the scaling function that gives the best approximation of the mapping. Continuity conditions are normally expressed in terms of the function spaces C^n, where n is the largest derivative that exists for the function. It seems intuitive that a function of order C^n should be best approximated by scaling functions of order C^n. For example, it would seem that a mapping that is only differentiable once, C^1, would be best approximated using a second order spline as the scaling function.

  The scaling function used in the approximation can be related to the continuity conditions of the mapping from the regualrization term in equation (3) derived in section 2 :

  $$\hat{P}PF(\mathbf{W}, \mathbf{x}) = \frac{1}{\lambda} \sum_{i=1}^{N} (f(\mathbf{x}) - F(\mathbf{W}, \mathbf{x}))\delta(\mathbf{x} - \mathbf{x}_i).$$

  The differential operator P in the above equation is set to the known continuity condition of the mapping. The Green's function solution to the differential equation is then calculated. Since scaling functions in the transform technique must be splines, the appropriate scaling function for the approximating function is the order of spline that most closely approximates the Green's function calculated.

- **Representation** : One of the key issues that this paper has not addressed is how data must be represented to achieve high network performance. A simple illustration of the importance of representation arose in pose estimation. For pose estimation the network was presented a mapping from a point in the input space (eigenspace) to $sin(\theta)$ and $cos(\theta)$ rather than θ. This greatly decreased the number of basis functions required to perform the mapping accurately. Since a function can be represented in a variety of ways, a methodology to evaluate representations is necessary. Using such a methodology, the appropriate representation for a function can be determined. This is helpful because the appropriate representation reduces the size of the network required to perform the mapping.

There exist many transforms, conformal mappings, transcendental mappings, and other techniques to represent a function. Suppose a function $f(x)$ can be represented in terms of any two of the above techniques, where $\tilde{f}(x)$ and $\bar{f}(x)$ are the two representations of $f(x)$. The question now arises, which representation of $f(x)$ is better ? Approximation spaces such as the Besov space offer possibilities [7] to answering the above question. For the representation $\tilde{f}(x)$ an error function :

$$\tilde{a}_N(\tilde{f}) = ||\tilde{f}(x) - \tilde{f}'(x)||^2$$

is defined where $\tilde{f}'(x)$ is a decomposition of $\tilde{f}(x)$ into N orthonormal basis. Given two representations with respective errors $\tilde{a}_N$ and $\bar{a}_N$, the class of functions $f(x)$ are cataloged for which $\tilde{a}_N$ and $\bar{a}_N$ decays at set rate as N gets large. A representation has a class α for the set of functions $f(x)$ that satisfy :

$$a_N = O(N^{-\alpha}).$$

The better representation is the one that for a given α contains a larger class of functions. It has been shown that the α class and Besov spaces are closely related [6] :

$$O(N^{-\alpha/2}) \Leftrightarrow f \in B_2^\alpha$$

when the error function is L^2. B_2^α is a Besov space. In addition, the relation between Besov spaces and continuity conditions of the class of functions in the space have been explored in approximation theory. It may be useful to examine these results.

- **Discrete Wavelet Transforms on Unevenly Sampled Data** : The issue of performing the wavelet transform on unevenly sampled data has been side stepped by using the Lomb periodogram. The assumptions made in using the periodogram and the errors introduced in going between continuous and discrete representations need to be explored. In future research, we hope to address the issues raised in these sections.

8 Appendix A

Theorem A.1 (*Unser, Aldroubi, Eden* 1992[13]) : Functions of the form $\beta^n(x)$, β-splines, tend to Gaussians as the order of the spline tends to infinity. A wavelet basis derived using using β splines as scaling functions approach the Gabor function as $n \mapsto \infty$:

$$\beta^n(x) \cong \sqrt{\frac{6}{\pi(n+1)}}\, e^{-\frac{6x^2}{n+1}} \tag{35}$$

This approximation of $\beta^n(x)$ satisfies both of Micchelli's conditions [10]. The approximation is asymptotic and can be shown to be highly accurate even for as low an order as $n = 3$. This is the rationale for using cubic β-splines as a basis functions in our RBF networks.

9 Appendix B

There exist several ways to construct wavelet bases using β-splines as scaling functions. The wavelet basis used in our work was the Battle-Lamarié orthonormal basis [9]. The scaling functions are cubic β-splines :

$$\beta^3(x) = \sum_{j=0}^{4} \frac{-1^j}{3!} \binom{4}{j} (x+2-j)^3 \, \mu(x+2-j)$$

where μ is the unit step function.

The filter formulas, $\overset{\circ}{v}(k)$ and $\overset{\circ}{w}(k)$, that were used in section 3.2 to calculate the wavelet coefficients, $d_{j,k}$, are given in Tables 3 and 4. These formulas were calculated by Unser, Aldroubi, and Eden [14] [15].

filter	frequency response
$\overset{\circ}{v}(k)$	$\frac{1}{2}U_2^3(f)\sqrt{\frac{B_1^7(f)}{B_1^7(2f)}}$
$\overset{\circ}{w}(k)$	$\frac{1}{2}e^{-2\pi f}U_2^3(f+\frac{1}{2})\sqrt{\frac{B_1^7(f+\frac{1}{2})}{B_1^7(2f)}}$

Table 3: The filter formulas for the transfer functions $\overset{\circ}{v}(k)$ and $\overset{\circ}{w}(k)$ [14] [15].

function	frequency response
$B_1^7(f)$	$\frac{1}{5040}(2416 + 1191[e^{2\pi f} + e^{-2\pi f}] + 120[e^{4\pi f} + e^{-4\pi f}] + e^{6\pi f} + e^{-6\pi f})$
$U_2^3(f)$	$\frac{1}{8}(e^{4\pi f} + 4e^{2\pi f} + 6 + 4e^{-2\pi f} + e^{4\pi f})$

Table 4: The components of the transfer functions $\overset{\circ}{v}(k)$ and $\overset{\circ}{w}(k)$ [14] [15].

References

[1] P. Baldi, "Computing with Arrays of Bell-Shaped and Sigmoidal Functions," *Advances in Neural Information Processing Systems,* R.P. Lippman, J.E. Moody, and D.S. Touretzky (ed.), Morgan Kaufman, pp. 735-742, 1991.

[2] M. D. Buhmann and C. A. Micchelli, "Spline Prewavelets for Non-uniform Knots," *Numerische Mathematik,* Vol. 61, pp. 455-474, 1992.

[3] M. D. Buhmann and C. A. Micchelli, "Spline Prewavelets for Non-uniform Knots," *Numerische Mathematik,* Vol. 61, pp. 455-474, 1992.

[4] M. D. Buhmann, "Multiquadratic Prewavelets on Non-Equally Spaced Centres," *Technical Report 94-06,* Seminar für Angewandte Mathematik Eidgenössische Technische Hochshule, July 1994.

[5] C. K. Chui, *An Introduction to Wavelets,* Academic Press, San Diego, 1992.

[6] R. A. Devore, B. Jawerth, and B. J. Lucier, "Image Compression Through Wavelet Transform Coding," *IEEE Transactions on Information Theory,* Vol. 38, No. 2, pp. 719-746, 1992.

[7] R. A. Devore, B. Jawerth, and V. A. Popov, "Compression of Wavelet Decompositions" *American Journal of Math,* 1992.

[8] N. R. Lomb, "Least-Squares Frequency Analysis of Unequally Spaced Data," *Astrophysics and Space Science,* Vol. 39, pp. 447-462, 1976.

[9] S. G. Mallat, "A Theory for Multiresolution Signal Decomposition: The Wavelet Representation," *IEEE Trans. on Pattern Analysis and Machine Intelligence,* Vol. 11, No. 7, pp. 674-693, 1989.

[10] C. A. Micchelli, "Interpolation of Scattered Data: Distance Matrices and Conditionally Positive Definite Functions," *Constructive Approximation,* Vol. 2, pp. 11-22, 1986.

[11] H. Murase and S. K. Nayar, "Learning and Recognition of 3D Objects from Appearance," *Proc. of IEEE Workshop on Qualitative Vision,* pp. 39-50, June 1993.

[12] T. Poggio and F. Girosi, "Networks for Approximation and Learning," *Proc. IEEE,* Vol. 78, pp. 1481-1497, 1990.

[13] M. Unser, A. Aldroubi, M. Eden, "On the Asymptotic Convergence of β-spline Wavelets to Gabor Functions," *IEEE Trans. on Information Theory,* Vol. 38, pp. 864-872, March 1992.

[14] M. Unser, A. Aldroubi, and M. Eden, "A Family of Polynomial Spline Wavelet Transforms," *Signal Processing,* Vol. 30, pp. 141-162, 1993.

[15] M. Unser, A. Aldroubi, M. Eden, "The L_2 Polynomial Spline Pyramid," *IEEE Trans. on Pattern Analysis and Machine Intelligence,* Vol. 15, No. 4, pp. 364-378, April 1993.