An Adaptive Filtering Interpolator Using Neural Networks

Zhou WANG and Yinglin YU
Research Institute of Electronics and Automation
South China University of Technology
Guangzhou 510641, P. R. China

ABSTRACT

Filtering interpolators presented by Lucke and Stocker[1] have advantages in reducing interpolation error in image background clutter-suppression systems especially for data with low sampling rates. Before they are to be applied, a fixed parameter a should be predetermined. We think if the parameter a is well adjusted, it may also be useful to recover an image from a less densely sampled image. Experiments show that interpolation error relies greatly on the parameter a and the best values of a for certain images are much different. Therefore, how to determine the values of a becomes the key problem for this application.

In this paper, we develop a neural network based adaptive system to automatically adjust the value of a. A modified robust BP algorithm is used in the training procedure for our special use. Simulation results show that a can be generated automatically by the neural networks instead of being blindly predetermined to a fixed value. Compared to the interpolator with best fixed parameter a, interpolation results are also improved.

1. INTRODUCTION

Filtering interpolators proposed by Lucke and Stocker [1] are originally used in image background clutter-suppression systems where a second image is subtracted from a first in a frame differencing signal processor (FDSP), in order to suppress the fixed background on a pixel-by-pixel basis. Since one frame must almost always be resampled before it is compared to another, interpolation error contributes to clutter leakage in the difference frame. Lucke and Stocker [1] show that filtering interpolators are powerful in reducing clutter leakage especially for data with low sampling rates. When a filtering interpolator (polynomial or trigonometric) is to be used, a fixed parameter a should be predetermined which can be chosen by users in their special uses.

Although the original use of filtering interpolator is for clutter-suppression processing, we think it may also be useful to recover an image from a less densely sampled image. This recovering procedure can be employed by an image compression/decompression system where the less densely sampled image can be viewed as the compressed codes and by applying the interpolator, we can obtain an interpolated image (decompressed image) from the less densely sampled image. Experiments show that the interpolation error is more sensitive to the parameter a in this process. Therefore, how to determine a becomes the key problem for this application.

To get better interpolation results, we are thinking of developing a system to adaptively adjust a for the local filtering interpolator where the inputs of the system are the value of the local sampling points and the output is a. This can be illustrated as a nonlinear function approximation problem. Multilayer feedforward neural networks have been proposed as a tool for nonlinear function approximation [2,3,4] and back propagation (BP) algorithm is a widely used learning algorithm for training multilayer networks. The training data for the networks are rather *fuzzy*. We describe it as *fuzzy* for two reasons. The first is that the desired outputs can be very different values even if the inputs are similar. The second is that their related interpolation mean square errors (MSE) change a lot in different cases. Standard BP algorithm is sensitive to these complex training data. Chen and Jain [5] proposed a way to reduce the influence of bad training data. In our special use, this algorithm can be improved by considering another parameter k, which is an approximate error sensitive measure for the training data.

In section 2, we will briefly describe the filtering interpolators. In section 3, a modified robust BP neural network is discussed. We introduce our adaptive system in section 4. Some simulation results are given in section 5 and concluding remarks are made in section 6.

2. FILTERING INTERPOLATION

A local one dimension interpolator can be described as the formula:

$$y_s = \sum_{n=-(N/2-1)}^{N/2} x_n h(n-s) \qquad (1)$$

where x_n denotes the values of samples, $h(x)$ is the interpolation function and y_s is the estimated value of the point at the position s. For images with two dimensions, the interpolator is applied to each dimensions separately. For a standard interpolator, $h(n) = \delta(n)$. When s=0, they simply reproduce the data to which they are applied. For $s \neq 0$, there will be some error when y_s is compared to x_s. Filtering interpolators are different from standard interpolators which use spectral filter to fit the sampling data and the condition $h(n) = \delta(n)$ need not necessarily be satisfied. Lucke and Stocker [1] developed parameteried families of filtering interpolators, one of them is the DFT-4 interpolator, whose interpolation function is

$$h(x) = \frac{1}{4}[1 + 2(1-2a)\cos(\pi x/2) + (1-4a)\cos(\pi x)] \quad (2)$$

the parameter a can be chosen by users.

Filtering interpolators have shown its power in reducing interpolation error for frame differencing signal processor (FDSP) where they are applied to both two data sets, one shifted and the other non-shifted. In this case, the evaluation standard for interpolation results is the error when the two interpolated data sets are compared.

We think if the parameter a is well adjusted, the filtering interpolators may also be used to recover an image from a less densely sampled image. In this case, the interpolator only applied on one data set and the evaluation standard for interpolation results is the error when the interpolated data compare to the original data. Experiments show that the interpolation results are very sensitive to the value of a in this application. The best fixed values of a change largely with data spectrum. We realize that if a can be adaptively adjusted, better interpolation results may be obtained.

3. THE NEURAL NETWORKS

As discussed in section 1, we can use multilayer feedforward neural networks to automatically adjust the parameter a for filtering interpolators. To get a set of training data, a 4 dimension vector $\mathbf{X}$ is randomly extracted from digital training images. The 4 components of the vector are consequently equally spaced 4 samples and their positions on the axis are fixed to be -1, 0, 1 and 2 respectively. The desired interpolation output is a vector $\mathbf{Y}_d$ whose components are the values of the points between 0 and 1 while the real interpolation output vector $\mathbf{Y}$ is generated by the filtering interpolator. We apply a 4 points discrete cosine transformation (DCT) to the input samples and discard the direct component which we think have nothing to do with the adequate value of a, and then we get a 3 dimension vector $\mathbf{X}_t$ which will be the input of the neural networks. This process can be illustrated as a feature extraction procedure. We also need a desired output of a (a_d). Given an input vector $\mathbf{X}$ and a parameter a, the filtering interpolator can generate an output vector $\mathbf{Y}$. a_d is which makes the mean square error (MSE) between $\mathbf{Y}$ and $\mathbf{Y}_d$ the least. We can get a_d by using a discrete form of Newton's method [6]. An example of MSE-a curve is shown in Fig.1. Experiments show that almost all MSE-a curves are much like 2-order polynomial curves but their curvatures are different. The curves with larger curvatures means their MSEs are more sensitive to a while those with smaller curvatures are less sensitive to a. To approximately measure the sensitivity, another parameter k is computed which is the difference between MSE correspond to a_d (MSE_d, the least value of MSE) and MSE correspond to a_d+Δa (MSE_c), where Δa is a constant (see Fig. 1).

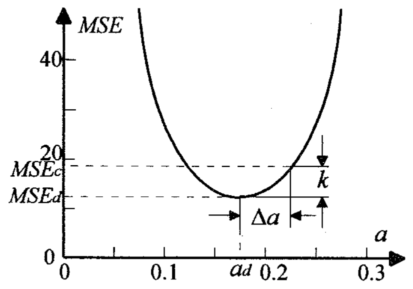

Fig .1 An example of MSE-a curve

A set of training data ($\mathbf{X}_t$, a_d, k) is employed by the BP algorithm to adjust the weights of the neural networks. The networks are composed of several independent subnetworks. A classification algorithm according to the magnitudes of $\mathbf{X}_t$ is used to determine which subnetwork will be trained. We use several subnetworks instead of one just to lessen the complexity of training data. By evaluating the quality of the training data, a robust learning algorithm was developed by Chen and Jain [5]. We think the evaluation can be more accurate by considering k and we develop a modified robust BP algorithm for our special use. To illustrate our algorithm briefly, we only discuss the difference between our algorithm and standard BP algorithm. In standard BP algorithm, the error on the output node which will be back propagated is :

$$\delta^p = (a_d^p - a_o^p) \quad (3)$$

where a_d^p and a_o^p are desired output and network output of the parameter a respectively, we modify (3) to :

$$\delta_R^p = g(1/k)\Psi_t(a_d^p - a_o^p) \tag{4}$$

The function *g(x)* (see Fig. 2) is defined as follows:

$$g(x) = \exp(-x^2 / c^2) \tag{5}$$

where C is a constant. The function $\Psi_t(r)$ (see Fig. 3) is the same as that in [5] :

$$\Psi_t(r) = \begin{cases} r & |r| \le a(t) \\ c_1 \tanh(c_2(b(t)-|r|))sign(r) & a(t) < r \le b(t) \\ 0 & |r| > b(t) \end{cases} \tag{6}$$

where $a(t)$ and $b(t)$ are time dependent cutoff points while c_1 and c_2 are constants. In standard BP algorithm, all training data are equally learned, while in our approach, to what extent a set of training data will be learned is determined by its quality. The set of training data with larger *k* (more important) and smaller $(a_d^p - a_o^p)$ (good training data) will be learned more, otherwise it will be learned less.

After training, the weights of the networks are stored and the neural networks can recommend us a value of a for any an input vector $\mathbf{X}_t$.

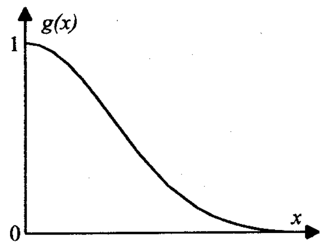

Fig. 2 The function of g(x)

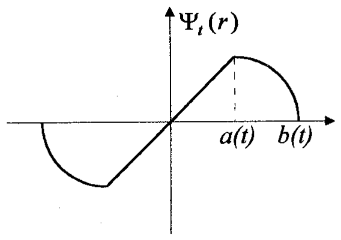

Fig. 3 The function of $\Psi_t(r)$

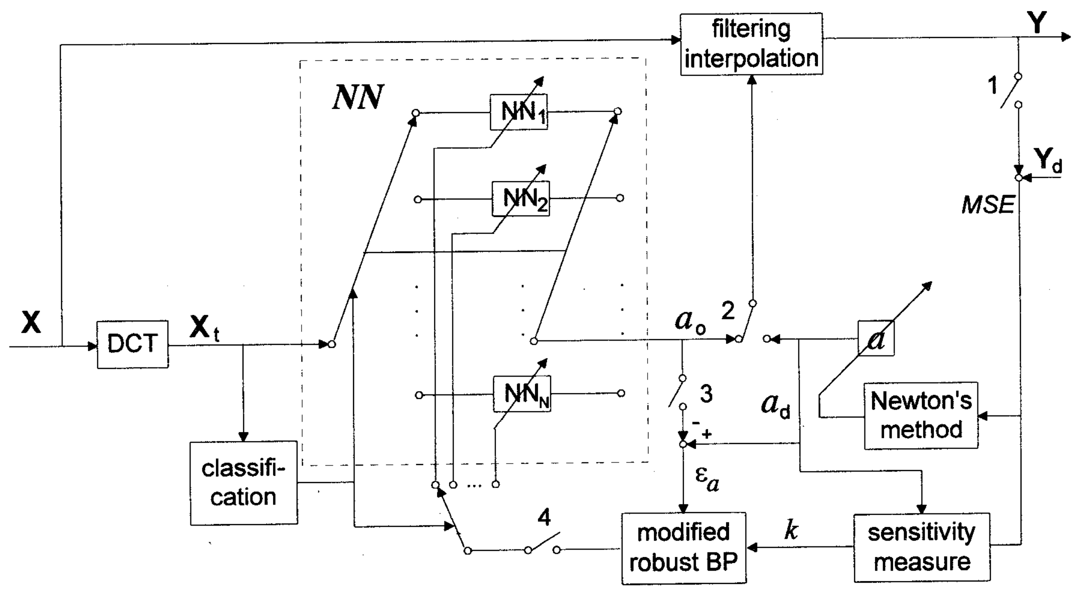

Fig. 4 The adaptive system (When training, switches 1,3,and 4 are closed and switch 2 is turned to the right side.)

4. THE ADAPTIVE SYSTEM

Our adaptive system is shown in Fig. 4.

When training, the switches 1,3 and 4 are closed and switch 2 is turned to the right side. In each training step, an input vector **X** and a desired system output vector $\mathbf{Y}_d$ which are randomly extracted from the training images are inputted to the system. $\mathbf{X}_t$ is generated by the DCT transform and the neural networks give an output a_o. Through Newton's method, we get a_d and then obtain sensitivity measure k. The difference between a_o and a_d ($\varepsilon_a = a_d - a_o$) and k are employed by the modified BP algorithm to modify the weights of one of the subnetworks. By iteratively apply the training step millions of times. The average ε_a decreases gradually.

When testing, the switches are turned to the other side (as shown in Fig. 4). A gray level digital image is less densely equal space resampled. Our adaptive interpolation system is applied to the sample data repeatedly horizontally and then vertically to generate an interpolated image. For each time, 4 consequent sample points are inputted to the system as the input vector **X**. $\mathbf{X}_t$ is computed and one of the subnetworks is activated by the classification algorithm to recommend us an a_o. a_o together with **X** are used by the filtering interpolator and the output is vector **Y** which is the interpolation data for the points between the 2nd and 3rd samples of **X**.

5. SIMULATIONS

The training and testing images we used are 512×512, 8bpp gray level images, some of which are shown in Fig. 5. To train our adaptive system, several of them are employed as training images. When training, we resample the testing images every 8 pixels in both horizontal and vertical directions and use DFT-4 interpolator with different fixed parameter a and our adaptive interpolator to generate interpolated images. To evaluate interpolation results, we use peak signal-to-noise ratio (PSNR) as the distortion measure.

$$PSNR = 10\log_{10}\frac{255^2}{\frac{1}{K}\sum_{i=1}^{K}(z_i - z_i')^2} \tag{7}$$

Where z_i and z_i' are the ith pixels from the original image and the interpolated image respectively, and K is the number of pixels in one image.

The best values for fixed parameter a are listed in Table 1 and *PSNR-a* curve for "LENA" with sampling rate every 8 pixels is shown in Fig. 6. It is clear that the best a for different images are different and PSNR will decline when a deviate the best a. Therefore, it is difficult to blindly predetermine a fixed value of a for a certain image.

Table 2 shows the interpolation results for testing image "LENA" with different training images. Compare to Table 1, we find that the interpolation results of our system are slightly better than those with best fixed a. PSNR does not change a lot with different training images, so we can conclude that the generalization ability of our system is good. We think this is due to the large amount of the training data and the classification algorithm in the adaptive system.

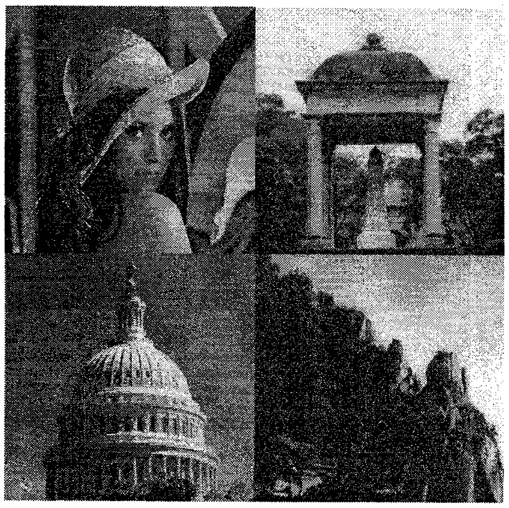
Fig. 5 Some of the training and testing images. (up-left: "LENA", up-right:"MB", down-left:"WH", down-right:"HS")

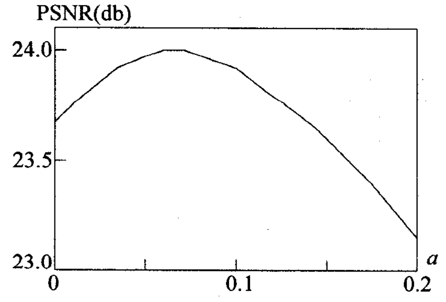

Fig. 6 An example of PSNR-a curve (image: "LENA", sampling rate: every 8 pixels)

Table 1 Best fixed a and related PSNR

image	sampling rate	best a	PSNR(db)
LENA	every 8 pixels	0.069	24.01
MB	every 8 pixels	0.059	25.69
WH	every 8 pixels	0.147	23.94
HS	every 8 pixels	0.100	26.08

Table 2 Adaptive interpolation results
(testing image: "LENA", sampling rate: every 8 pixels)

training image(s)	PSNR(db)
LENA	24.08
LENA, MB, WH, HS	24.07
MB, WH, HS	24.06
10 images (not including LENA)	24.07

6. CONCLUSIONS

We develop an adaptive filtering interpolation system to recover an image from a less densely sampled image. In our system, the parameter a for filtering interpolation can be automatically generated by the neural networks instead of blindly predetermined to a fixed value. The interpolation results are slightly better than those with the best fixed values of a. The generalization ability is to some extent good. These advantages make our system more practical in real applications.

To train the feedforward neural networks employed by our adaptive system, standard BP algorithm is not adequate because of the complexity of the training data. A robust BP algorithm[5] can make good training data being learned more while bad training data being learned less. By considering another parameter k, we develop a modified robust BP algorithm. This modification makes more important training data (with larger k) being learned more and less important training data (with smaller k) being learned less.

Our system may be improved in several respects:

The DCT transformation applied on the input vector **X** is somewhat a feature extraction procedure. Further research about the features of sampling data and interpolation performances may give us better ways to extract features and classify the network input data.

We find it is difficult to determine c in (5), $a(t)$, $b(t)$, c_1 and c_2 in (6) and k in Fig.1 is also an approximate estimation. Improvements in obtaining these parameters may improve the efficiency of the network learning algorithm.

Other kinds of neural networks may replace the BP networks in our system.

Filtering interpolator may be more useful when dealing with smoother images while our training and testing images have too many detail regions which are difficult for any interpolation algorithm. Therefore, more smoother images should be involved in the training and testing procedure.

All our training and testing data are extracted from gray level images. How our system performs when dealing with other kinds of signals and how about the generalization ability between different kinds of signals could be studied.

ACKNOWLEDGMENT

This work is supported partly by China National Natural Science Foundation and partly by China National "Climbing" project.

REFERENCES

[1] R. L. Lucke, A. D. Stocker, Filtering interpolators for frame differencing signal processors, IEEE Trans. on Signal Processing, vol. 41, no. 8, 1993.
[2] K. Hornik, M. Stinchcombe, H. White, Multilayer feedforward networks are universal approximators, Neural Networks, vol. 2, 1989.
[3] C. Ji, R. R. Snapp, D. Psaltis, Generalizing smoothness constraints from discrete samples, Neural Computation, vol. 2, 1990.
[4] T. Poggio, F. Girosi, Networks for approximation and learning, Proc. IEEE, vol. 78, no. 9, 1990.
[5] D. S. Chen, R. C. Jain, A robust back propagation learning algorithm for function approximation, IEEE Trans. on Neural Networks, vol. 5, no. 3, 1994.
[6] B. Widrow, S. D. Stearns, Adaptive signal processing, Prentic-Hall, Inc., 1985.