ABSTRACT

COMPUTER SCIENCE

HALL, KEVIN D. B.S. CLARK ATLANTA UNIVERSITY, 1996

HYBRID NEURAL NETWORK IMAGE PROCESSING TESTBED

Advisor: Dr. Kenneth Perry

Thesis Dated July, 1996

The focus of this research was to establish a testbed for pattern recognition. In this testbed, the wavelet transform is used as a preprocessor for various neural networks. The wavelet transform is used to perform image compression, and several wavelet filters and compression techniques are implemented. The compressed data is later formatted and used as input to a neural network where pattern recognition is performed.

The wavelet filters used in the wavelet transformation were the Daubechies 4 (DAUD4) and the Haar wavelet filters. After compression was performed, the root mean square error (RMS) was computed and compared with a "common" compression technique called JPEG compression. After testing each compression technique, zone compression using the wavelet transform yielded the best results. At this point, the compressed data was used by various neural networks for pattern recognition.

There were three neural nets in the testbed. They were the neocognitron, a genetic algorithm driven neural network, and the Hopfield neural net. Each neural net was used to perform pattern recognition using the compressed data. The results from each neural net were good, but the neocognitron gave the best results.

HYBRID NEURAL NETWORK IMAGE PROCESSING TESTBED

A THESIS

SUBMITTED TO THE FACULTY OF CLARK ATLANTA UNIVERSITY

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR

THE DEGREE OF MASTER OF SCIENCE

BY

KEVIN D. HALL

DEPARTMENT OF COMPUTER SCIENCE

ATLANTA, GEORGIA

JULY 1996

Rix T 78

ACKNOWLEDGMENTS

I would like to thank Dr. Kenneth Perry for introducing me to the area of image compression and guiding me through this research. Also, I am very grateful for the assistance of Dr. Hsin-Chu Chen.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

Figure

LIST OF ABBREVIATIONS

DAUD4	**Daubechies Wavelet 4**
DCT	**Discrete Cosine Transform**
DWT	**Discrete Wavelet Transform**
GAs	**Genetic Algorithms**
JPEG	**Joint Photographic Experts Group**
PC	**Personal Computer**
RMS	**Root Mean Square Error**

CHAPTER 1

INTRODUCTION

This thesis presents research in the areas of pattern recognition and image compression. Pattern Recognition is computer recognition of stimulus patterns[6]. Although this seems to be a simple task for humans, recognizing similar features in patterns is a difficult task for a computer. This is due to the fact that most methods for pattern recognition are oversensitive to shifts in position and the distortions in shape of the stimulus patterns.

In this research, a "testbed" was developed to perform pattern recognition. This "testbed" consisted of a preprocessor and various networks. The preprocessor uses the wavelet transform to perform image compression. This compression algorithm takes a pgm image and compresses it. The image is next converted into a bitmap format, and this data is feed into a neural network. The neural networks are used to perform pattern recognition.

Pattern recognition has been an area of research for many years. A common problem with performing pattern recognition was not being able to recognize an image after a shift in position or distortion in shape. A solution to this problem was later established. Kunihiko Fukushima proposed a new algorithm for pattern recognition which is tolerant of deformations and shifts in position[6]. The algorithm is known as the neocognitron.

Neocognitron

The neocognitron is an improved version of the conventional cognitron. Its self-organization is performed by unsupervised learning, a "learning-without-a-teacher" process[5]. However, the cognitron did not have the ability to recognize stimulus patterns when there were distortions or shifts in position.

The self-organization of the neocognitron is also performed by unsupervised learning. The repeated presentation of a set of stimulus patterns is necessary for the self-organization of the neocognitron[6]. No information about the categories to which these patterns should be classified is needed. Unlike the cognitron, the neocognitron acquires the ability to classify and correctly recognize these patterns by itself. Recognition is done with respect to differences in shapes. Most of all, the neocognitron recognizes stimulus patterns correctly without being effected by shifts in position or even by considerable distortions in shape of the stimulus patterns.

The structure of the neocognitron is hierarchical. The information of the stimulus pattern given to the input layer of the neocognitron is processed step by step. This takes place in each stage of the network, and a cell in a deeper stage responds selectively to the complex feature of the stimulus pattern. At the same time, the cells in a deeper stage has a larger area in which information is transmitted from the stimulus pattern to the cell. Thus, each cell in the deepest stage responds only to a specific stimulus pattern without being affected by the position or the size of the stimulus pattern. This classification process can become even more difficult if the stimulus pattern consists of compressed image data.

Image Compression

Image Compression is a technique used to reduce the number of bits required to represent an image. It allows an image to retain most of its features, but the image is composed of fewer pixels. Image compression is very valuable because it facilitates the storage and transmission of an image. When storage space is limited, compression is often essential for space conservation.

Another use of image compression is in transmitting images. An image can contain a large amount of data, and sending a large image to a secondary location can be very time consuming. A compressed image allows easier transmission of the image data from place to place because less data is being transmitted. The fast transmission of image data can be very crucial, for example, medical and military applications.

Image compression techniques can be divided into two major families; lossy and lossless[11]. In this research, lossy image compression was implemented. Lossy image compression concedes a certain loss of accuracy in exchange for greatly increased compression[10]. Most techniques for lossy image compression can be adjusted to different quality levels, and higher accuracy is gained in exchange for less effective compression. As a result, a trade-off between accuracy and compression is one of the major factors that must be considered in the development of this compression algorithm.

Various techniques are used to perform lossy image compression. In this research the wavelet transform was used as a method of image compression.

Wavelets

Wavelets are a recent development in the area of applied mathematics. They originated over ten years ago from the works of various scientists and engineers. Among these scientists and engineers were Morlet, Arens, Fourgeau and Giard(1982), Morlet(1983), and Grossmann and Morlet(1984). Ingrid Daubechies was another scientist that has done much work using wavelets. In the past few years, researchers of various backgrounds have developed a strong interest in wavelets. Wavelets have already lead to exciting applications in signal analysis (sounds, images) and numerical analysis (integral transform). The one thing that contributes to the strong interest in wavelets is their "wide applicability."

Wavelets are mathematical functions, and there are many types of wavelets. These mathematical functions are represented by $f(x) = \Sigma b_{jk} W(2^j x-k)$ in which b_{jk} carries information about f near $\xi = 2^j$ and $x = 2^{-jk}$. Wavelets are based on two indices in which k is translation ($W(x) \to W(x + 1)$) and j is dilation or compression. Dilation plays a very important role in the construction of wavelets. The basic dilation equation is a two-scale difference equation represented by $\Phi(x) = \Sigma c_k \Phi(2x-k)$. W is defined as the wavelet that is derived from the scaling function Φ by taking the differences: $W(x) = \Sigma(-1)^k c_{1-k} \Phi(2x-k)$. The term c_k is defined as the wavelet coefficients.

There are infinitely many possible sets of wavelets. Wavelets are often categorized by the numeral values called wavelet filter coefficients. Each set of wavelets offers different trade-offs. The most distinguished trade-offs are between how compactly they are localized in space and how smooth they are.

In this thesis, we restricted ourselves to the Daubechies and Haar wavelet filters. Daubechies has a class of wavelet filters, and we chose the filter called DAUD4. DAUD4 is known as the simplest and most localized member of the class except Haar. This particular wavelet filter has only four coefficients, and they are displayed in table 1. The equations for the unknown coefficients were first recognized and solved by Daubechies[13].

Table 1. DAUD4 coefficients and their solutions

Coefficients	C_0	C_1	C_2	C_3
Equations	$((1 + \surd 3) / 4\surd 2)$	$((3 + \surd 3) / 4\surd 2)$	$((3 - \surd 3) / 4\surd 2)$	$((1 - \surd 3) / 4\surd 2)$

The purpose of these coefficients is to construct the transformation matrix. The transformation matrix is a square matrix, and it acts on a column vector of data. Though the size of the matrix differs, most transformation matrices are constructed like the one in Figure 1.

$$\begin{bmatrix} C_0 & C_1 & C_2 & C_3 & & & & \\ C_3 & -C_2 & C_1 & -C_0 & & & & \\ & & C_0 & C_1 & C_2 & C_3 & & \\ & & C_3 & -C_2 & C_1 & -C_0 & & \\ \vdots & \vdots & & & \ddots & & & \\ & & & & C_0 & C_1 & C_2 & C_3 \\ & & & & C_3 & -C_2 & C_1 & -C_0 \\ C_2 & C_3 & & & & & C_0 & C_1 \\ C_1 & -C_0 & & & & & C_3 & -C_2 \end{bmatrix}$$

Fig. 1. The general format of the transformation matrix.

In Figure 1, the blank spaces signify zeroes. The number of zeroes needed is determined by the length of the column vector. The first row in the matrix generates one component of the data "convolved with the filter coefficients $C_0, \ldots, C_3$."
Like the first row, every odd number row performs the same task. The even number row performs a different convolution with the coefficients C_3 $-C_2$, C_1, $-C_0$. The wavelet filter with coefficients $C_0, \ldots, C_3$ is defined as the smoothing filter, and it outputs the "smooth" information. The filter with coefficients C_3, $-C_2$, C_1, $-C_0$, does the opposite. This filter produces the data's "detail" or high frequency information.

To make the technique of transforming data useful, there must be a way to reconstruct the original data from the transformed data. This requires the transformation matrix in Figure 1 to be orthogonal. If the transformation matrix is orthogonal, its inverse is the transposed matrix. The transpose of the transformation matrix is displayed in Figure 2.

$$\left[\begin{array}{cccccccc}
C_0 & C_3 & & & \cdots & & C_2 & C_1 \\
C_1 & -C_2 & & & \cdots & & C_3 & -C_0 \\
C_2 & C_1 & C_0 & C_3 & & & & \\
C_3 & -C_0 & C_1 & -C_2 & & & & \\
 & & & \ddots & & & & \\
 & & C_2 & C_1 & C_0 & C_3 & & \\
 & & C_3 & -C_0 & C_1 & -C_2 & & \\
 & & & & C_2 & C_1 & C_0 & C_3 \\
 & & & & C_3 & -C_0 & C_1 & -C_2
\end{array}\right]$$

Fig. 2. The transpose of the transformation matrix.

The Discrete Wavelet Transform (DWT) is a linear and fast operation that is performed on a data vector whose length is of integer power two. DWT transforms this data vector into a different vector of the same length. Like the fast Fourier Transform, DWT can be seen in a function space, from the input space domain to the wavelet domain[13]. In the input space domain, the basis functions are the unit vectors e_i, to the wavelet domain. In the wavelet domain, the basis functions are more complex and they are called "mother functions" and "wavelets."

Unlike the sines and cosines in the fast Fourier Transform, individual wavelet functions are quite localized in space and frequency. The characteristic of dual localization given by the wavelet functions yields large classes of functions and operators sparse to some high accuracy, when transformed into the wavelet domain.

The DWT consists of applying a wavelet coefficient matrix like Figure 1. This operation is done hierarchically. First, the wavelet coefficient matrix is applied to the full data vector of length *N*, then to the "smooth" vector of length *N*/2. Next, it is applied to the "smooth-smooth" vector of length *N*/4, and so on until only a trivial number of "smooth-...-smooth" components (usually 2) remain[]. This process is known as the pyramidal algorithm. The DWT yields output that consists of the remaining components and all the "detail" components. This is illustrated in Figure 3.

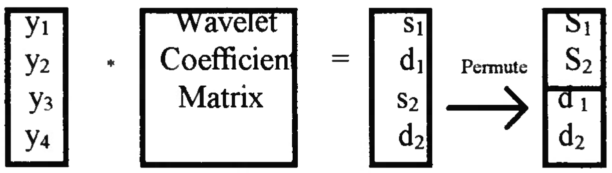


Fig. 3. Displays pyramidal procedure

The wavelet transform of any data vector depends on two important factors. The length of the data vector is one of those factors. If the length of the data vector $(y_1, ..., y_n)$

were a higher power of two, there would be more stages of applying the wavelet coefficient matrix (Figure 1) and permuting. The number of wavelet coefficients is the other important factor. If the wavelet coefficient matrix contains many coefficients, naturally, there are going to be more stages in this procedure. This procedure continues until the endpoint consists of a vector with two *S*s and a hierarchy of *D*'s, D's, d's, etc. The Ss are the smooth data, and the D's are the detail or high frequency data. Once d's are generated, they simply propagate through to all subsequent stages. To invert the DWT, the procedure is reversed. Using the transformation matrix in Figure 2, the inverse procedure starts with the smallest level of the hierarchy to the highest.

CHAPTER 2

RESEARCH METHODOLOGY

In this research, the approach taken was to first construct an image compression "testbed" using the wavelet transformation. This image compression system is to be able to: (1) input an image, (2) compress the image, (3) reconstruct the image, and (4) convert image formats. The second phase was to perform pattern recognition using various neural networks. In performing these tasks, several design issues were solved.

Design Issues

An initial design issue was the image format. The image format was an issue because the content of the image needed to be displayed. Also the image data needed to be abstracted and manipulated. Most of all, the image needed to be of a format that could be reconstructed after compression. The image format was also an issue in performing pattern recognition. The image needed to be in a format in which it could be converted to ones and zeros. This particular format was necessary to input data for the neural network. Another design issue was the wavelet filter.

The wavelet filter was an issue because of the variety of filters. When choosing a wavelet filter, the trade-offs mentioned in Chapter 1 had to be considered. These trade-offs were between how compactly they are localized in space and how smooth they are.

The most important factor about choosing a wavelet filter was whether it would produce good compression results. The final design issue dealt with selecting the types of neural network to include in the "testbed". The type of neural network to use to perform the pattern recognition was another design issue. The neural network must have the ability to properly classified the sample patterns consisting of compressed data. An important factor that also had to be considered was whether the image data could be converted to run with the neural network.

Approach For Resolving Design Issues

Resolutions to the above design issues were the results of much research in the various areas. The design issue of the image format was resolved by testing various image formats. These tests were based on the following factors: (1) can the image contents be edited, (2) can the image data be manipulated, (3) can the image data be abstracted from the image, and (4) can the image be reconstructed. The appropriate image format for the image compression algorithm and the pattern recognition process was chosen based on the above factors. The design resolution for the wavelet filter took a similar process.

The process of choosing a wavelet filter that would produce good results was difficult. The characteristics needed in a wavelet filter were: (1) must be localized in time, and (2) produce good results. The method used to find a suitable filter was "trial-and-error." Several wavelet filters were tested, and the filters that yielded the best results was chosen. The final design resolution was determining the type of neural networks to perform pattern recognition.

As stated in Chapter 1, pattern recognition is a technique that enables patterns of similar features to be categorized in the same group or class. This classification is to be performed by a neural network. Finding a network to perform such a task was very difficult. To resolve this design issue, the decision was made to construct a “testbed” of various neural networks. By creating this “testbed," the results could be compared.

CHAPTER 3
SYSTEM DESIGN

The first task in this research was to design an image compression algorithm using wavelets. The initial stage in creating this algorithm is to determine the types of image format to use. In the next section, a description of the image format is given.

Description of Image Format

The type of image format used in this research design was pgm. The image was composed of 256 possible gray scales. The structure of this image format is displayed below in Figure 4.

```
 P2
 # CREATOR: XV Version 3.00  Rev: 3/30/93
 8 8
 255
255 255 255 255 255 255 255 255
255 255 255 255 255 255 255 255
  0    0 255 255 255 255 0    0
  0    0 255 255 255 255 0    0
  0    0 255 255 255 255 0    0
  0    0 255 255 255 255 0    0
255 255 255 255 255 255 255 255
255 255 255 255 255 255 255 255
```

Fig. 4. Displays the structure of a pgm image file

This image format was chosen because of the following features: (1) it was easy to understand, (2) the image data can be easily abstracted, and (3) the image can be reconstructed.

When using wavelets to perform any type of transformation, a wavelet filter must be used. In the next section a description of the wavelet filter is given.

Description Of Wavelet Filter

As stated previously, a particular set of wavelets is categorized by the wavelet filter. In this research, the particular wavelet filter used was developed by Daubechies[2]. The wavelet filter is called DAUD4. DAUD4 is a wavelet filter that has four coefficients. These coefficients are displayed in Table 1. These coefficients are used to construct the matrix that is applied to the data vector.

When the wavelet coefficient matrix is applied to the data vector, a transformation takes place. The process converts the image data into a numerically difference image of the same size. Compression takes place after the wavelet transformation is done. In this research, there are three compression techniques used. In the following section, these compression techniques are discussed.

Compression Techniques

As stated in Chapter 1, compression is reducing the number of bits required to represent an image. There are two types of compression: (1) lossy and (2) lossless. In this research, lossy compression was implemented. The compression techniques performed in this research were thresholding, zone compression, and Joint Photographic Experts Group (JPEG). These compression techniques are discussed in the following sections.

Wavelet Transform

As mentioned in the introduction, the wavelet transform is a linear mathematical operation that is performed on a data vector whose length is an integer power two. The wavelet transform does not perform any form of compression, but compression techniques can be done after the transformation. In this thesis, two compression techniques were used to compress the transformed data. In the next sections, these techniques discussed in more detail.

Thresholding

Two compression techniques were performed on the transformed image data created by the wavelet transformation. The first technique was thresholding. Thresholding is a compression technique in which a value or threshold is set, and the items of the image data are kept or replaced based on the threshold value. In other words, the integer values in the image data are kept if they are equal to or greater than the threshold value. If the integer values are less than the threshold value, the integer values are replaced by zero. This is illustrated in Figure 5.

225 204 115 234 44	Threshold Value 200	225 204 0 234 0

Fig. 5. Illustrates thresholding as a form of compression

Zone Compression

Another compression technique used after the wavelet transformation is done is zone compression. Zone compression is a compression technique in which compression is done by selecting a region of the transformed data. This is illustrated below in Figure 6.

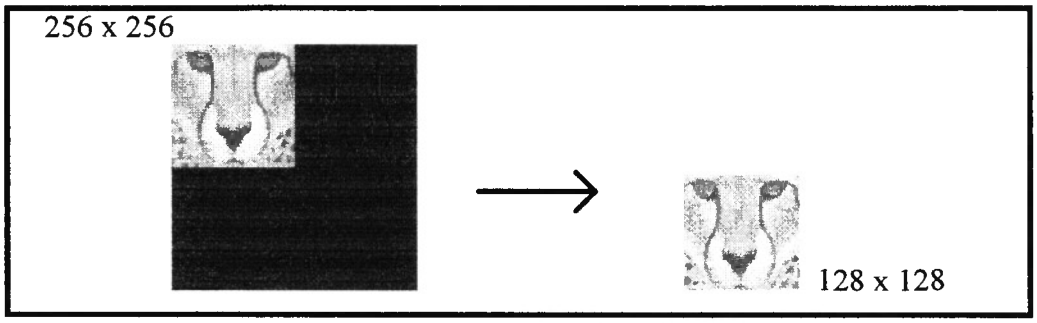


Fig. 6. Illustrates zone compression after performing the wavelet transformation

This compression technique was a valuable technique used to extract the low frequency data from the transformed image data. The low frequency data was extracted because it contains the most information, and in the following chapters, this compression technique is shown to be most effective.

The final compression technique used was JPEG. This compression algorithm was used for the purpose of comparison, and it is discussed in the next section.

JPEG

JPEG is a compression technique that was developed several years ago, and it is often considered the "standard" compression technique[11]. JPEG is an acronym that stands for Joint Photographic Experts Group. Like the above compression techniques, JPEG is also a lossy compression algorithm. This compression algorithm operates in three successive stages.

The discrete cosine transform (DCT) is the first stage. The DCT is in a class of mathematical operations that includes the well-known Fast Fourier Transform (FFT), as well as many others[13]. The basic operation performed by these transforms is to take the image data and transform it from one type of representation to another. The DCT is closely related to the Fourier Transform, and often yields similar results.

The second stage is quantization. The DCT output matrix takes more space to store than the original matrix of pixels[11]. In this stage, the process of quantization reduces the number of bits needed to store the pixel values. This is done via a quantization matrix. For every element position in the DCT matrix, a corresponding value is generated in the quantization matrix. The quantum value generated indicates what the step size is going to be for that element in the compressed rendition of the picture, with values ranging from one to 255.

The final step in the JPEG process is coding the quantized images. The JPEG coding phase combines three different steps to compress the image. The first changes the DC coefficient at 0,0 from an absolute value to a relative value. Next, the coefficients of the image are arranged in the "zig-zag sequence." This "zig-zag sequence" is used to compress the consecutive zero value produced in the quantization stage.

After compression, the compressed data is formatted, and the formatted data is used as input patterns for the following networks. The process of pattern recognition is performed using various neural networks. In the next section, these neural networks are described.

Description Of Neural Networks

Artificial neural networks are information-processing systems that have certain performance characteristics in common with biological neural networks[4]. These neural networks were developed as generalizations of mathematical models of human cognition or neural biology. Artificial neural networks are characterized by (1) its pattern of connections between the neurons, (2) its method of determining the weights on the connections, and (3) its activation function. In the following section, the neural networks used in this research is discussed.

Neocognitron

The neocognitron is an algorithm developed by Kunihiko Fukushima. The purpose of the neocognitron was to recognize patterns correctly without being affected by shifts in position or distortions in shape of the sample patterns. In this research, a C-implementation of Fukushima's neocognitron was used. This program was developed by Frank Schnorrenberg. Frank Schnorrenberg was a student in the Computer Science Department at Texas A&M University. This algorithm was developed by Schnorrenberg in 1992, and it was made available for public use on the Web.

The neocognitron developed by Schnorrenberg has many characteristics similar to the neocognitron developed by Fukushima, but there were a few changes. Among the things that was similar was that the network is unsupervised. When a neural network is unsupervised, training of the network takes place with repeated presentation of the patterns.

There were also variations from the original neocognitron algorithm developed by Fukushima. One difference was the lack of V_c cells. V_c cells are inhibitory cells, whose output terminals are connected only to inhibitory input terminals of other cells. Another different was the number of planes in the various stages. The differences are displayed in Table 2.

Table 2. Displays the number of planes per stage

	U_0	U_1	U_2	U_3	U_4
Schnorrenberg	1	20	20	20	20
Fukushima	1	24	24	24	24

From the above table, the number of planes per stage differ. Although there were differences in Schnorrenberg's implementation, the patterns tested with the program were properly classified.

Hopfield Neural Network

The Hopfield Neural Net is a fully interconnected neural network, in the sense that each unit is connected to every other unit[4]. The net weights are symmetric with no self-connections. This means $w_{ij} = w_{ji}$, and $w_{ii} = 0$. In the Hopfield Neural Net only one unit updates its activation at a time. Each unit continues to receive an external signal, and a signal from the other units is also received. The updating of the units allows a function known as the energy function to be found for the network. This function provides proof that the network can converge to a stable set of, activation instead of oscillating. Besides guaranteed convergence, the most importance features of this neural network are the asynchronous update of the weights and the zero weights on the diagonal.

Another form of neural network used was a feed-forward neural network that was trained using a genetic algorithm.

Genetic Algorithm Trained Feed-Forward Neural Network

A feed-forward neural net is a type of network in such the input units travel directly to the output units. In this research, a multilayer feed-forward neural net with six hidden layers was used. To train the neural net, a genetic algorithm was used.

Genetic algorithms (GAs) are a class of randomized search procedures capable of adaptive and robust search over a wide range of search space topologies[]. GAs implement a very powerful form of hill climbing technique that guards against local minimal. Using a genetic algorithm to train a feed-forward neural network was the thesis research of Samuel Collins. This research actually stemmed from the Pascal implementation of a Simple Genetic Algorithm program created by Goldberg (1989). In this research, the genetic algorithm used crossover and mutation to perform the training of the image data. Crossover is an operation that takes two candidate solutions and divides them, swapping components to produce two new candidates. Mutation is another operation in which a single candidate is taken, and some aspects are randomly changed in the candidate. Using these two operations, the feed-forward neural net was trained by the genetic algorithm, and the image data was entered for classification.

CHAPTER 4

IMPLEMENTATION

The first stage in this implementation process was to develop an algorithm to perform image compression. The method used to perform the compression was (1) perform the wavelet transform of the image and (2) apply compression techniques to the transformed data. In the following section, the implementation of image compression using the wavelet transform is discussed.

Wavelet Transform

As mentioned in chapter 1, the wavelet transform is linear operation that is applied to a data vector of the power two. The data vector is then transformed into a numerically different vector of the same length. The data used in this research was actual images.

Input File

The type of image file format used in this research was pgm. The images used were of type gray scale, and the size of these images was 256 x 256. As mentioned in the previous chapter, this type of image format was used because (1) image content can be viewed, (2) data can be manipulated, and (3) image can be reconstructed. Reading the image into the program was the first step.

When reading the image into the program, several things take place. The user was first prompted for the image filename, and the file was opened. Once the file is opened, the file contents are read, and the data is decomposed into three parts. These three parts were (1) file header, (2) dimensions of image, and (3) image data. In Figure 5, this is implemented..

```
printf("Enter Image filename \n");
scanf("%s", filename);

if((fpfile = fopen(filename,"r")) == NULL) {
        printf("\nERROR: Could not open \n");
         exit(-1); }

if((fpmat = fopen("data/matrixdata","w")) == NULL){
          printf("\nERROR: Could not open %s\n", "data/matrixdata");
          exit(-1);}

if((fphead = fopen("data/header","w")) == NULL){
          printf("\nERROR: Could not open %s\n", "data/header");
          exit(-1);}

if((fphead = fopen("data/header","w")) == NULL){
          printf("\nERROR: Could not open %s\n", "data/header");
          exit(-1);}

count = 1;
while(!feof(fpfile)){
fgets(headoffile, 126, fpfile);
if (count <= 4)
    fprintf(fphead, "%s", headoffile);
if (count == 3) fprintf(fpdim, "%s", headoffile);
if (count > 4)
    fprintf(fpmat, "%s", headoffile);
 count++;
  }
```

Fig. 7. Implementation for inputting the image file

At this point, the image is decomposed into the necessary components, and the wavelet transform can be performed on the image data stored in the file "matrixdata."

Wavelet Transform Procedure

The wavelet transformation involved applying the a wavelet filter to the image data. The wavelet filter used in this research was the DAUD4. This procedure is illustrated in Figure 8. The image data was first read from the file "matrixdata" into a two-dimensional array of type double. The wavelet transform was initially applied to the wavelet filter row-by-row. At this point each row of the image was read into a one-dimensional array, and this array was pass into the wavelet filter (DAUD4). The process was also performed for each column of the image data. Each time the transform was performed the results were stored back into the two-dimensional array.

```
DAUD4(double a[], unsigned long n) {
    double *w;       unsigned long i, j, k, half, vhalf, m, p;

    w = vector(1, n);
    half = n/2;
    vhalf = half + 1;
    for(i = 1, j = 1; j <= n-3; i++){
        w[i] = C0*a[j] + C1*a[j+1] + C2*a[j+2] + C3*a[j+3];
        w[i + half] = C3*a[j] - C2*a[j+1] + C1*a[j+2] -C0*a[j+3];
        j = j + 2;}
    w[i] = C0*a[n-1] + C1*a[n] + C2*a[1] + C3*a[2];
    w[i + half] = C3*a[n-1] - C2*a[n] + C1*a[1] - C0*a[2]; }}

    for(k = 1; k <= n; k++)   a[k] = w[k]; }  /*  End Function */
```

Fig. 8. Implementation for DAUD4 wavelet filter

When the wavelet transformation was done on both the rows and columns, the results were stored into a file called "storage." The results stored in this file consisted both of positive and negative integers. To display this image data, the data must be all positive integers, and these integers must be in the range from 0 to 255. This means the data must be scaled to meet this requirement, and in the following section, the scaling function is discussed.

Scaling Function

The scaling function was used to create a scale positive integers ranging from 0 to 255. Scaling the data produced by the wavelet transform was necessary to view the results. To perform this operation, the smallest and largest integers in the transformed data were determined. These values were determined with the use of a quick sort algorithm, and they were returned by the procedure "qs." Afterwards, the scaling function was applied to every element in transformed data. This is illustrated in Figure 9.

```
value = qs(item, 1, number_of_samples, reval);
 smallest = value[0];  largest  = value[1];
 difference  = f - e;
tcb = (double *) malloc (number_of_samples * sizeof(double));
for (cd = 1; cd <= number_of_samples; cd++){
    tcb[cd] = (((input[cd] - smallest)/difference)* 255);
    fprintf(flp, "%3.0f ", tcb[cd] );
    if ( rt == ndimx){ fprintf(flp, "\n"); rt = 1;}
  else    rt++;}
```

Fig. 9. Implementation for scaling function

The scaling data was then written to a file known as "inverse.pgm." This file consisted of the reconstructed image after the wavelet transform was performed.

As mentioned in the previous chapters, two compression techniques were used after the wavelet transformation was performed. The first form of compression used was zone compression.

Zone Compression

Zone compression was a technique in which compression took place by selecting a region of the transformed image data. This technique was performed using the following implementation fragment.

```
    fpdata = fopen("data/matrixdata", "r");
    m = 1;
    n = 1;
    td = 1;
    for(i = 1; i<= elements; i++){
       fscanf(fpdata, "%d", &num[i]);
       if((m <= cdimx) && (n <= cdimy))
          fprintf(fpfinal, "%4d ", num[i]);
       if (m == dimx) m = 1;
        else
       m++;
       if(td == dimy){td = 1; fprintf(fpfinal, "\n"); n++;}
      else
       td++;

i++;
 }
```

Fig. 10. Implementation of zone compression

With the above fragment, the transformed data was read into an array, and if the data was in the proper region, it was used to construct the compressed image. Another compression technique used with the wavelet transform was thresholding.

Thresholding

Thresholding was another compression in which elements of the transformed data were kept or assigned a zero value depending on a threshold value. This technique is illustrated in Figure 5. To implement this compression technique, the following fragment was used.

```
 fpdata = fopen("data/matrixdata", "r");

printf("ENTER A THRESHOLD VALUE (0 - 255) \n");
scanf("%d", &threshold);

n = 1;
td = 1;
m = 1;

for(i = 1; i<= elements; i++){
    fscanf(fpdata, "%d", &num[i]);

    if (num[i] >= threshold)
        fprintf(fpfinal2, "%3d ", num[i]);
    if (num[i] < threshold)
        fprintf(fpfinal2, " 0 ");
 }
```

Fig. 11. Implementation of thresholding

Compression is often used for storage and transmission of data. Even though compression can be very valuable, it is sometimes desirable to reconstruct the original data.

Inverse Wavelet Transformation

The inverse wavelet transformation was used to restore the transformed data into its original data. In this research, the inverse wavelet transform was performed after compression was done. The type of compression performed was lossy compression, and the inverse process would not yield a perfect reconstruction. The inverse data would be similar to the original data. The inverse transform is computed by the following implementation fragment.

```
DAUD4(double a[], unsigned long n){
double *w;
unsigned long i, j, k, half, vhalf, m, p;

w = vector(1, n);
half = n/2;
vhalf = half + 1;
w[1]=C2*a[half]+C1*a[n]+C0*a[1]+C3*a[vhalf];
w[2]=C3*a[half]-C0*a[n]+C1*a[1]-C2*a[vhalf];

for(m=1,p=3; m < half; m++){
   w[p++]=C2*a[m]+C1*a[m+half]+C0*a[m+1]+C3*a[m+vhalf];
   w[p++]=C3*a[m]-C0*a[m+half]+C1*a[m+1]-C2*a[m+vhalf];}}
for(k = 1; k <= n; k++)
   a[k] = w[k]; } /*  End Function */
```

Fig. 12. Implementation of inverse wavelet transformation

The inverse transformation is a reverse operation of the wavelet transformation. The differences in this procedure are the wavelet filter matrix and the reversed process. In the inverse transformation the wavelet filter matrix is the transpose of the original transformation matrix. The end results of the inverse transform are data elements similar to the original image data.

The wavelet transformation and the two compression techniques could easily be performed using the following interface. This interface is displayed by typing "test" in the respected directory.

```
PROMPT>>> test

Enter the desired operation
(1)    Wavelet Transformation
(2)    Compression (Zone and Thresholding)
(3)    Error (RMS)
(4)    Quit
```

Fig. 13. Displays interface for wavelet compression program

This interface was designed to assist the user in operating this program, and getting results. The final compression technique used was the JPEG compression.

JPEG Compression

JPEG is a "standard" compression technique often used. In this research, JPEG was used for comparison, and it helped determine how effective the compression techniques performed with the wavelet transformation were. This implementation was taken from "The Data Compression Book", which was written by Mark Nelson and Jean-Loup Gailly. The algorithm is a C implementation that is compiled and executed on a personal computer (PC).

This program actually consists of two parts: (1) compression and (2) expansion. The compression was performed entering the following: main-c inputfilename outputfilename qualityvariable. The result is a compressed image, but the image can not be viewed. To view the image, the image must be expanded. The image is expanded by the following statement: main-e inputfile outputfile. The inputfile is the outputfile from the compression operation, and the outputfile is the file name for the reconstructed image. At this point, the image can be viewed using the following command: gs outputfile. To determine the mathematical difference between the reconstructed image and the original image, an error function was developed.

Error Function

The error function was developed to determine the error after the various compression techniques were administrated. In Figure 14, the original and the compressed files are opened.

```
if((fpmat1 = fopen("data/matrixfile1","r")) == NULL){
 printf("\nERROR: Could not open %s\n", "data/matrixfile1");
 exit(-1);}
 if((fpmat2 = fopen("data/matrixfile2","r")) == NULL){
printf("\nERROR: Could not open %s\n", "data/matrixfile2");
exit(-1); }
```

Fig. 14. Implementation for opening the two files

```
for(a =1; a <= dimx; a++){
    for(b = 1; b <= dimy; b++){
        fscanf(fpmat1, "%d", &form[a][b]);
        total = total + form[a][b]; }}
 fclose(fpmat1);
 for(c =1; c <= dimx; c++){
 for(d = 1; d <= dimy; d++){
     fscanf(fpmat2, "%d", &form2[c][d]);}}
 fclose(fpmat2);
 error = 0.0;

for ( m = 1; m <= dimx; m++ ) {
    for ( n = 1 ; n <= dimy; n++ ) {
        diff = form[m][n] - form2[m][n];
        error = error + (diff*diff); }}
error = error / total;
printf( "RMS error between is %f\n",  sqrt(error ) ); }
```

Fig. 15. Implementation fragment for error function

In Figure 15, the data of the original image (fpmat1) and the compressed image (fpmat2) were read into two-dimensional arrays. The data elements in the original image were added, and the difference between the two images was determined. The error was then computed and displayed. After the error was computed, the next major step was to perform pattern recognition. Pattern recognition was computer recognition of stimulus patterns. In this research, networks were used to perform pattern recognition using the compressed image and the original image.

Neural Networks

The neural networks used to do pattern recognition were hopfield, neocognitron, and a genetic algorithm driven network. Several neural networks were used to test the ability of different neural network to classified patterns containing compressed data.

Hopfield Neural Network

The Hopfield Neural Net is a fully connected single layer network that associates a pattern with stored patterns. This algorithm was implemented by Khalil Khalif and later modified by Samuel Collins. This neural net was designed to take an "exemplarfile" and a noisy pattern. Figure 13 illustrates a sample "exemplarfile" and noisy pattern.

```
Exemplarfile                                Noisy Pattern
 2  ------> Number of patterns               1 -1  1 -1
 4  ------> X-dimension of patterns          1 -1  1  1
 4  ------> Y-dimension of patterns          1 -1 -1 -1
                                            -1  1 -1 -1
-1  1 -1  1
-1  1 -1  1
-1  1 -1  1
-1  1 -1  1

-1 -1  1 -1
 1 -1  1 -1
 1 -1  1 -1
 1 -1 -1 -1
```

Fig. 16. Input parameters for the hopfield neural network

The "exemplarfile" is a file that consists of the following data: (1) number of patterns, (2) number of rows in each pattern, (3) number of columns in each pattern, and (4) the actual patterns. In this research, these patterns consisted of images of the size 32 x32. These images were of type xbm, and they were converted to ones and negative ones. The noisy pattern was also of type xbm and size 32 x32, but this file consisted of the compressed image data.

The Hopfield Neural Net is activated by the following expression: hopfield exemplarfile noisyfile. The program becomes executing, and it executes until the program converges. Convergence means a pattern in the "exemplarfile" is categorized as having the most features similar to the "noisyfile". The end results were the "exemplarfile" pattern with similar features and the energy. The neocognitron was another neural net used to perform pattern recognition.

Neocognitron

As mentioned in Chapter 1, the neocognitron was designed to properly categorize similar patterns even if there were shifts in position or distortion. The neocognitron used in this research was designed and implemented by Frank Schnorrenberg. This algorithm was a C implementation of Fukushima's neocognitron. Schnorrenberg designed this neocognitron simulator be very flexible by allowing command line options.

The usage of this neocognitron simulator program is easily defined by just typing "neo" at the prompt, and the various options are displayed. Although there are many options, there are some mandatory parameters. The simulator must be provided with three files for input. The first file is the file containing the patterns. The second file is the file containing all the weights. The final file contains the parameters α, r_1, q_1, and the specifications for the three planes that are monitored during the runs. The simulator is activated by typing "neo" and its parameters.

```
Usage:          neo -fin=<fn> -samp=# -fwgt=<fn> -use=# -fspec=<fn> -fout=<fn>
        -maxiter=# -reps=#

Example:     Let the network read and classify 10 sample pattern

      neo -fin=myinput.pat -samp=10 -fwgt=wgts.dat -fspec=specs.dat -maxiter=100
          -reps=5
```

Fig. 17. Usage of neocognitron

In the Figure 17, the usage of the neocognitron simulator was defined, and many parameters were given. In Table 3, these parameters are vividly defined.

Table 3. The parameters for the neocognitron is defined.

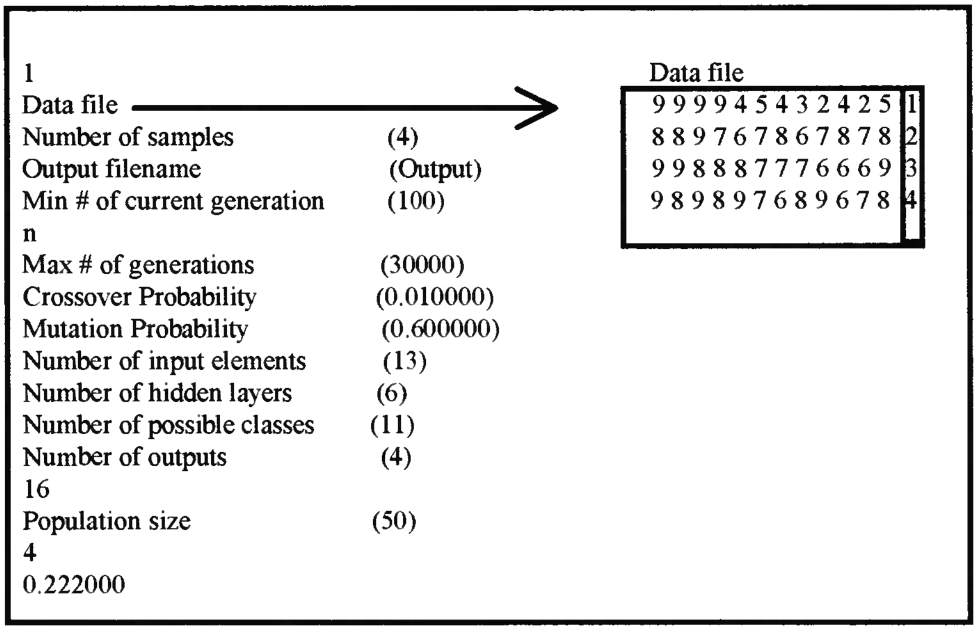
Parameters	Description
-fin=	name of train file containing the input pattern
-samp=	number of train input patterns in train file
-fwgt=	name of the file with weights)
-use=	use the weight file or not (1, 0)
-fspec=	name of the file with initial parameters
-fout=	name of the file to write output to
-maxiter=	maximal number of iterations
-reps=	number of repetitions for each pattern before proceeding

The command to activate the neocognitron is very long, but it gives the user the advantage of being able to change the variables of the simulator without having to edit the program. The final network used in this research was a feed-forward neural network.

Feed-forward Neural Network

This feed-forward neural net was designed and implemented by Samuel Collins as his thesis research. The unique feature about this neural net was that a genetic algorithm was used to train the feed-forward neural network. The implementation of this neural net was also made very flexible. This is due to the fact that the parameters of the programs can be change easily and without editing the program.

The input variable for this neural network is the parameter file. The parameter file was consisted of the data needed to train the neural network via the genetic algorithm. A sample parameter file is illustrated in Figure 14.



```
1                                                Data file
Data file  ------------------------------->     9 9 9 9 4 5 4 3 2 4 2 5 | 1
Number of samples            (4)                 8 8 9 7 6 7 8 6 7 8 7 8 | 2
Output filename              (Output)            9 9 8 8 8 7 7 7 6 6 6 9 | 3
Min # of current generation  (100)               9 8 9 8 9 7 6 8 9 6 7 8 | 4
n
Max # of generations         (30000)
Crossover Probability        (0.010000)
Mutation Probability         (0.600000)
Number of input elements     (13)
Number of hidden layers      (6)
Number of possible classes   (11)
Number of outputs            (4)
16
Population size              (50)
4
0.222000
```

Fig. 18. Structure of parameter file

Figure 18 displays the many things that must be defined. The data file was one component of the parameter file that required more attention. The data file contained the image data, and the image data had to be formatted. Also, the image data required an additional variable to define its class. This additional variable was very important because it was the item used to determine the classified of the different images. After these neural networks were tested the results were gathered.

CHAPTER 5

SUMMARY/CONCLUSION

A "testbed" was created, and a novel configuration of wavelet transformation and the neocognitron was explored. In the "testbed", image compression techniques were designed and implemented using the wavelet transform. The wavelet transformation was performed, and two compression techniques were done on the transformed data. The compression techniques used were thresholding and zone compression. To determine the efficiency of these compression techniques, two images were used to test the program. These compression techniques were also compared to JPEG compression.

"Cheetah.pgm" was the first image used to test these compression techniques. This image was of type gray scales, and it had the size of 256x256. This image is displayed in Figure 19.

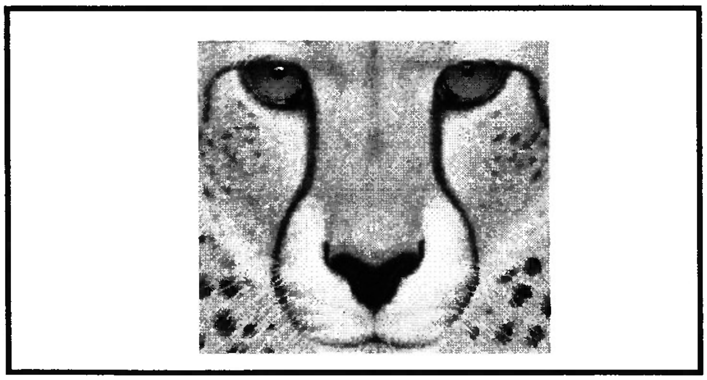


Fig. 19. Image used in testing compression techniques

The wavelet transform was performed, and thresholding was the first compression technique done on the transformed data. The same image was next compressed used the JPEG compression algorithm. To compare the results of both tests, the root mean square error (RMS) was computed for each compression ratio. In Table 4, the results from the two compression techniques are displayed.

Table 4. Results from thresholding and JPEG compression techniques

Cheetah		
Percentage	**JPEG**	**Wavelet**
		(Thresholding)
9	**0.289**	**4.54**
57	**4.289**	**3.45**
83	**9.884**	**2.19**
90	**14.460**	**2.70**
94	**22.025**	**6.41**

The results showed that the JPEG compression technique had the smaller error in the initial stages, but as the percentage of data withheld increased, the RMS for the JPEG compression technique also increased greatly. There was an increase in errror from the thresholding technique, but the degree of increase was not as large as the JPEG compression technique.

Although the RMS was a valuable way of comparing the two compression techniques, visual comparisons were also important. After comparing the actual images, the images compressed using JPEG were very good in the when the percentage was below 60 percent. As the percentage passed 60 percent, the images became very “blocky” in appearance. The images from the thresholding technique were also good in the initial stages, and as the percentage increased, the images began to darken. These images are displayed in appendix.

The second compression technique performed on the data transformed by the wavelet transformation was zone compression. As mentioned in the above chapters, zone compression was a technique in which a region of the transformed data was selected. The low frequency data (smooth data) was chosen in the zone compression technique. The image used in this technique was also “Cheetah.pgm” (Figure 19). The results are displayed in Table 5.

Table 5. Results from zone compression and JPEG compression techniques

Cheetah

Percentage	JPEG	Wavelet (Zone Compression)
75	6.88	1.86
94	22.025	2.68
98	56.04	3.90
99	59.88	4.79

The JPEG technique did not perform well at very high percentages, but the zone compression technique produced good results. These compression techniques were also used to compress another sample image.

To further test these compression algorithms, another image was include in this testing phase. The image used in the second testing phase was known as “lisa.pgm." This image was also of type gray scales, and its size was 256x256. The wavelet transformation was first performed, and thresholding and JPEG compression were done on the image.

The results from these compression techniques were also similar to the first sample input image. The error for the JPEG compression technique again increased greatly as the percentage of data withheld increased. This is shown in Table 6.

Table 6. Results from thresholding and JPEG compression techniques

Lisa		
Percentage	**JPEG**	**Wavelet (Thresholding)**
83	**2.32**	**4.07**
90	**3.19**	**3.53**
92	**3.76**	**3.47**
94	**7.22**	**3.49**
95	**8.57**	**3.90**

Zone compression was also performed, and the results were also compared with the results of the JPEG compression. Like the first image, the compression techniques using the wavelet transform produce the best results in all tests. Table 7 displays the results of the zone compression techniques.

Table 7. Results from zone compression and JPEG compression techniques

Lisa		
Percentage	**JPEG**	**Wavelet (Zone)**
75	**1.83**	**2.76**
94	**7.22**	**3.80**
98	**46.18**	**4.84**
99	**54.29**	**5.19**

In the testing of both images, the compression techniques used after the wavelet transformation was performed on the data yielded the best results. The zone compression technique seemed to produce the best overall results. The next phase was to do pattern recognition used the compressed data.

Pattern recognition is a simply being able to recognize and classify all samples with similar features. Pattern recognition was a done by various neural networks: (1) hopfield, (2) neocognitron, and (3) feed-forward neural network.

The first neural network used for pattern recognition was the hopfield neural net. The hopfield input files were the "exemplarfile" and the "noisyfile" (Figure 16). The noisyfile consists of one image pattern with compressed data, and the "exemplarfile" contains various image patterns with the original data. Given the compressed data, the Hopfield neural net was able to match it with the original data in the "exemplarfile." The results are illustrated in table 8.

Table 8. Results from Hopfield neural network

Exemplarfile		Hopfield Neural Network Noisyfile	Results	
No. of Patterns	Size	Percentage Compression	Was Pattern Recognized	No. of Iterations
2	32x32	75% of data withheld	yes	1
2	32x32	94% of data withheld	yes	2
3	32x32	94% of data withheld	yes	2
5	32x32	94% of data withheld	yes	4

The Hopfield Neural Network produced good results, but the number of iterations needed for convergence increased as the number of patterns increased in size. The next neural net used for pattern recognition was a feed-forward neural network.

As mentioned in the previous chapter, this neural net was trained by a genetic algorithm. The parameters used to train the genetic algorithm are displayed in Table 9.

Table 9. Parameters for the simple genetic algorithm (SGA)

Total Population size	50
Chromosome length (lchrom)	12600
Maximum # of generations (maxgen)	30000
Crossover probability (pcross)	0.010000
Mutation probability (pmutation)	0.600000
Number of neural net inputs (NNI)	100
Number of neural net hidden layers (NNHL)	6
Number of neural net outputs (NNO)	4
Number of weights (NNHL * ((NNI+1)+NNO))	630

The neural net was trained by the genetic algorithm, and the best results given were a classification of 33.33 percent and a fitness of 0.414214. The parameters were changed, but the fitness still remained at 0.414214, and the percentage of correct classification always remained at 33.33 percent. This was probably due to the fact that the image data was too large. The final neural net used to perform pattern recognition was the neocognitron.

The neocognitron was a neural net developed by Fukushima to perform pattern recognition. In this research, an implementation of Fukushima's neocognitron was used, and this implementation was developed by Frank Schonorrenberg. The results from the neocognitron were very good. The neocognitron was able to recognize and categorize the original image data and the compressed data.

The neocognitron was tested with ten (10) sample patterns. To train the neocognitron these sample patterns were presented to neocognitron twenty times. In the initial testing stage, the compressed data was not classified by the neocognitron. This was due to inadequate training. Therefore the number of repetitions was increased, and this caused the patterns to be presented to the neocognitron several more times. These changes ultimately lead to better training and good results.

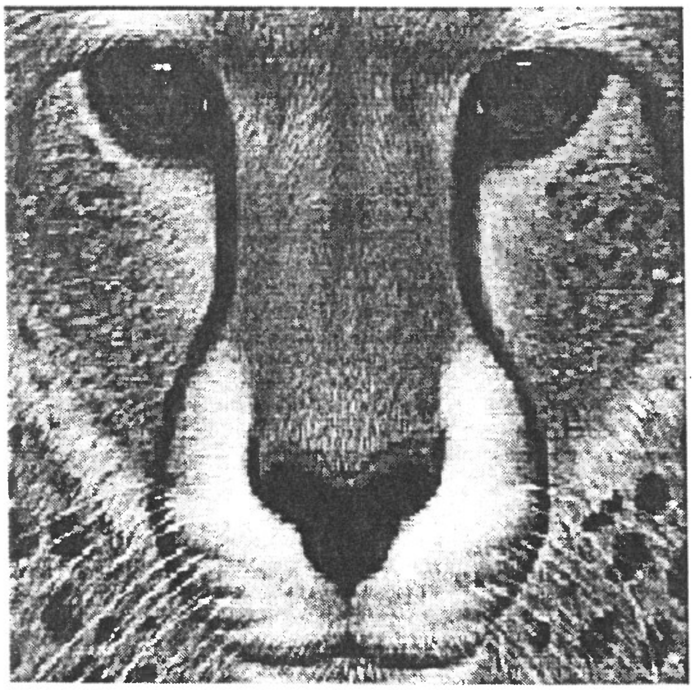
In this research, we were able to investigate the use of the wavelet transformation as a preprocessor to neural networks. The wavelet transformation was used for compression and feature extractions. Most of all, we were able to receive very good results from the neocognitron. The neocognitron was able to recognize the compressed patterns.

As a result of this research, several areas of further research were established. These areas of future work include: (1) adding other neural networks, (2) adding other wavelet filters, and (3) running the compression and neural neural network algorithms in real-time. These areas are future research for Lanier Watkins and Tracey Abrams.

APPENDIX 1
RESULTS FROM JPEG COMPRESSION

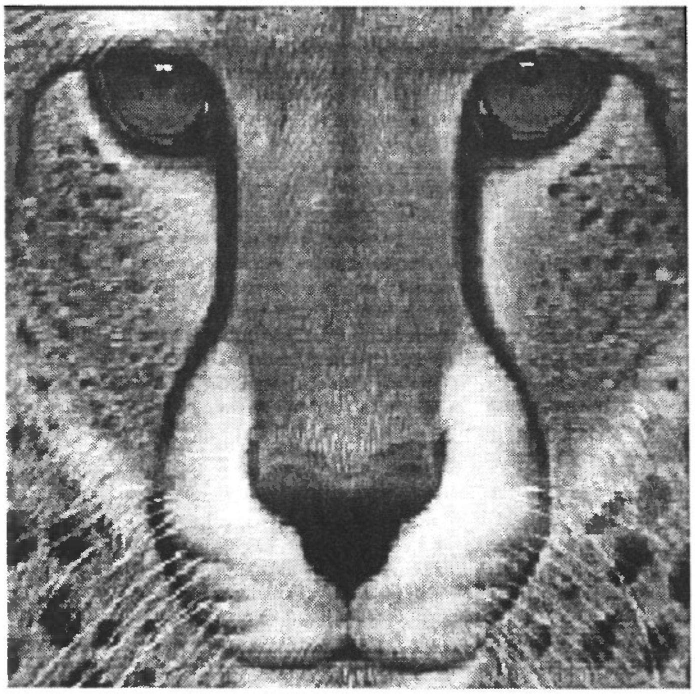
9% Compression (256x256)

RMS = 0.289



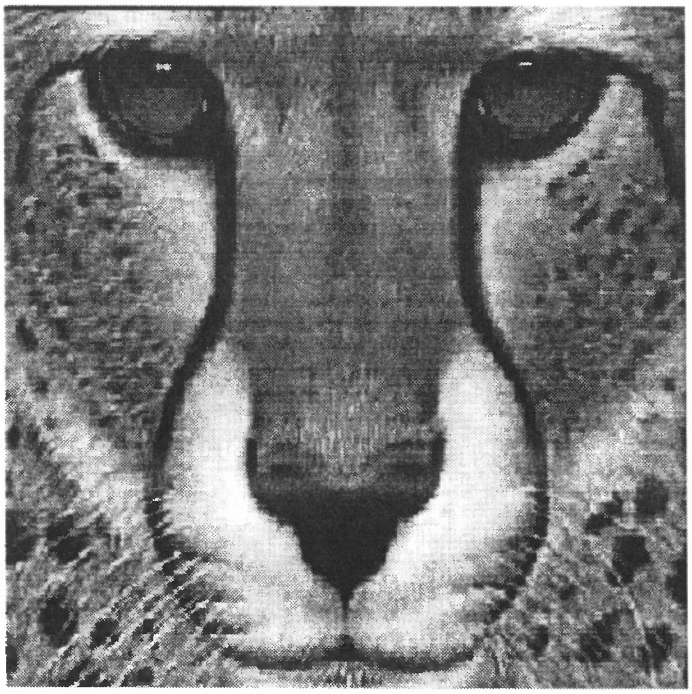
57% Compression (256x256)

RMS = 4.289



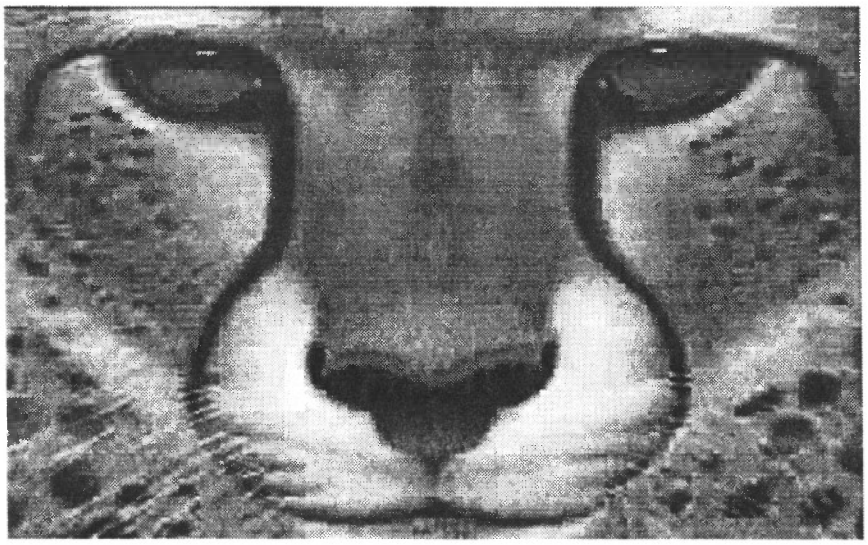
75% Compression (256x256)

RMS = 6.88



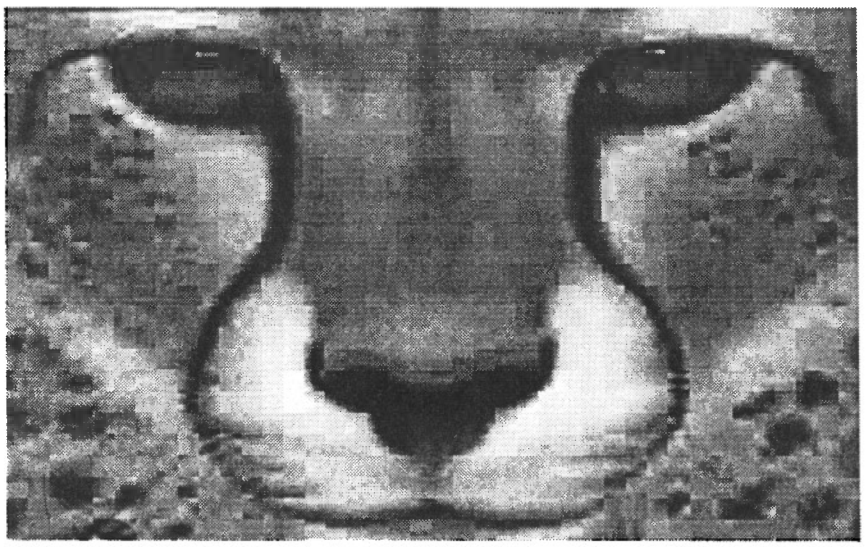
83% Compression (256x256)

RMS = 9.884



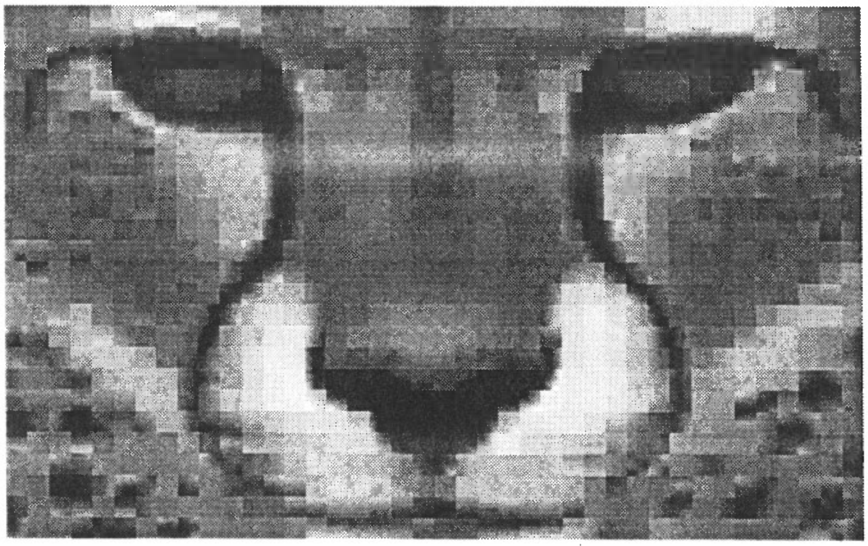
90% Compression (256x256)

RMS = 14.46



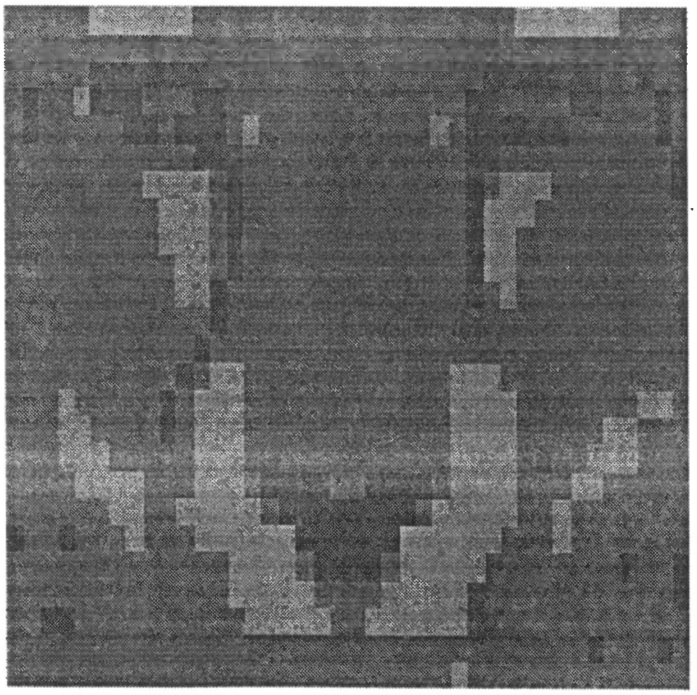
94% Compression (256x256)

RMS = 22.025



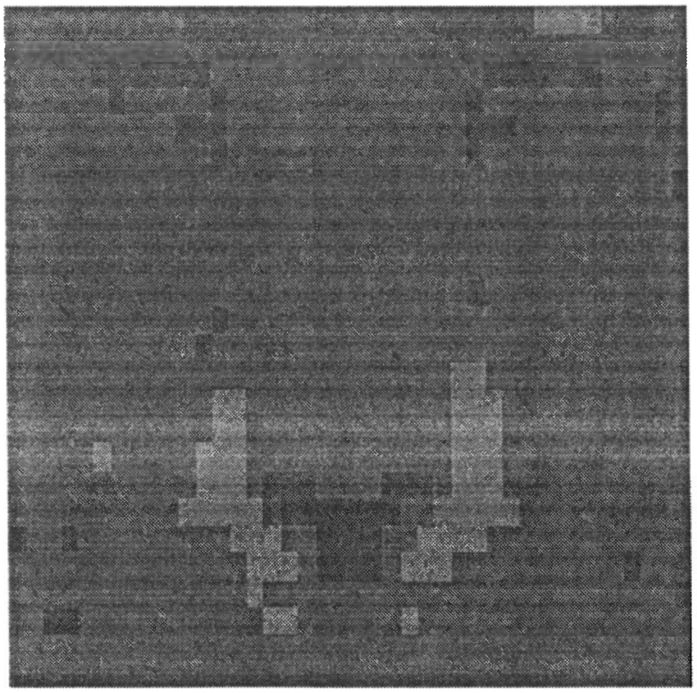
98% Compression (256x256)

RMS = 56.04



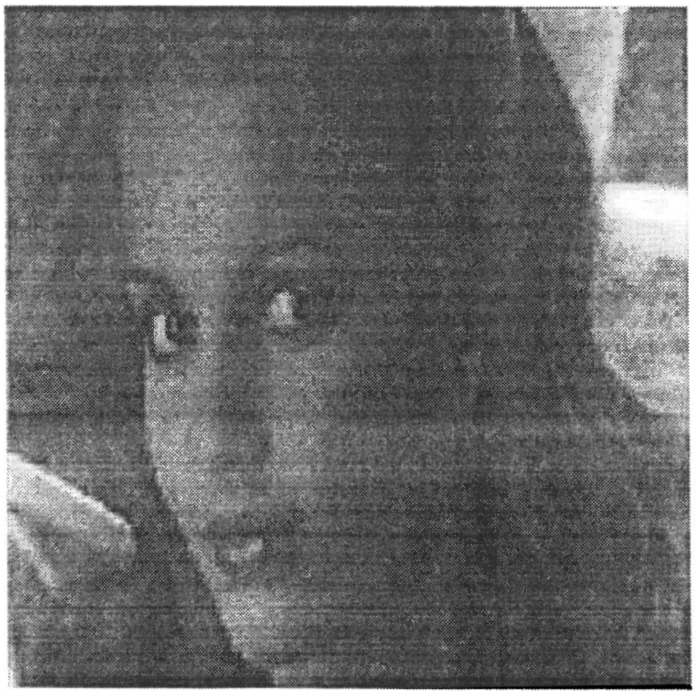
99% Compression (256x256)

RMS = 59.88



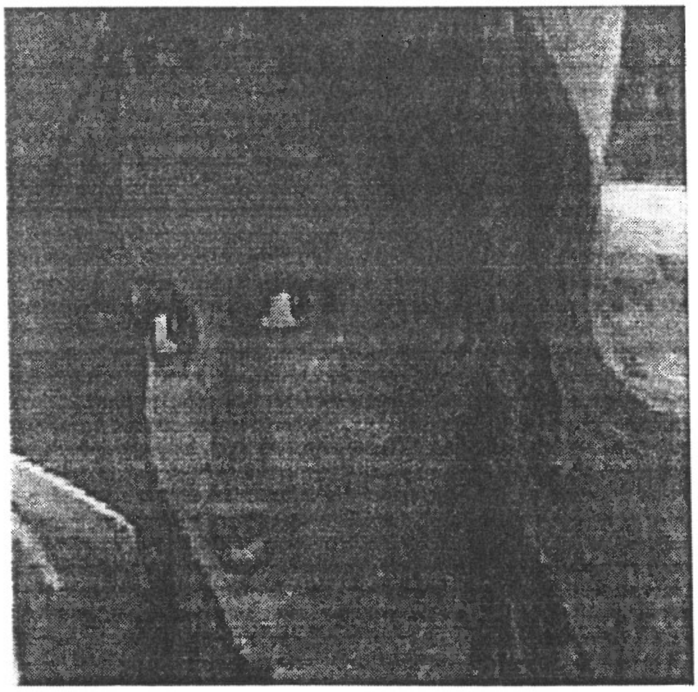
75% Compression (256x256)

RMS = 1.83



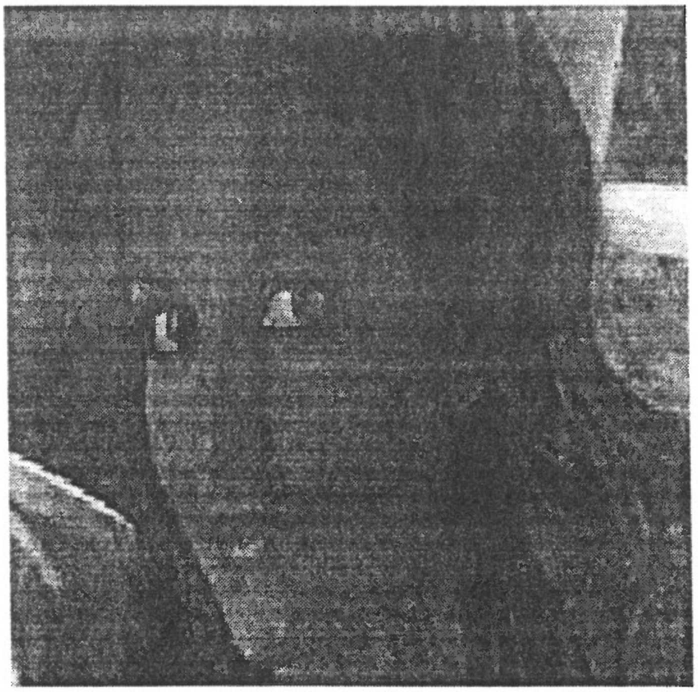
83% Compression (256x256)

RMS = 2.32



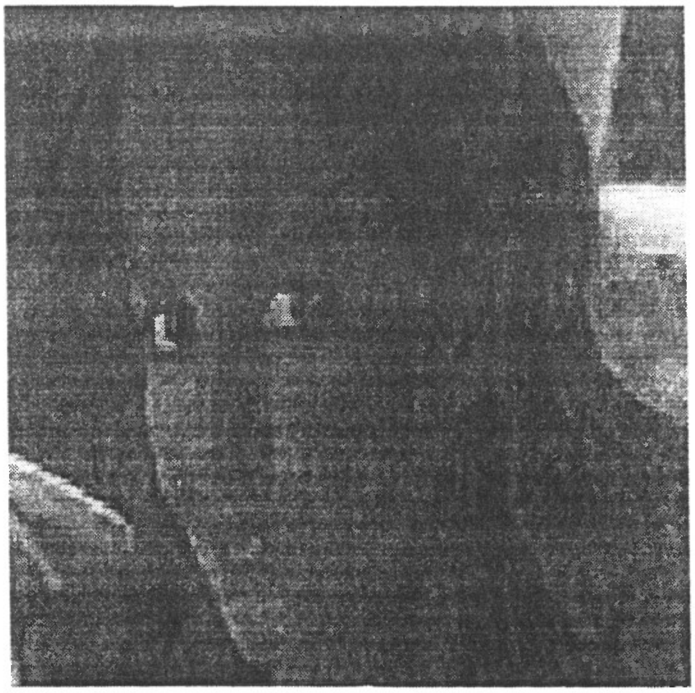
90% Compression (256x256)

RMS = 3.19



92% Compression (256x256)

RMS = 3.76

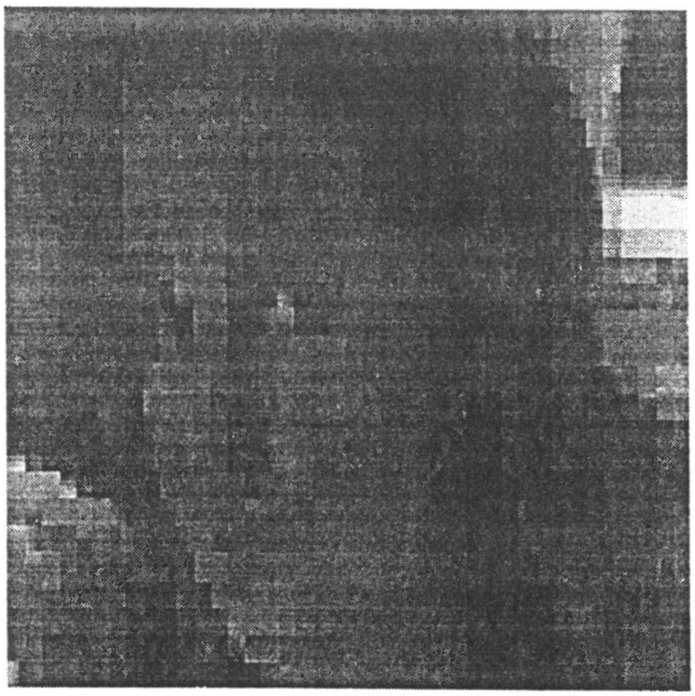


94% Compression (256x256)

RMS = 7.22

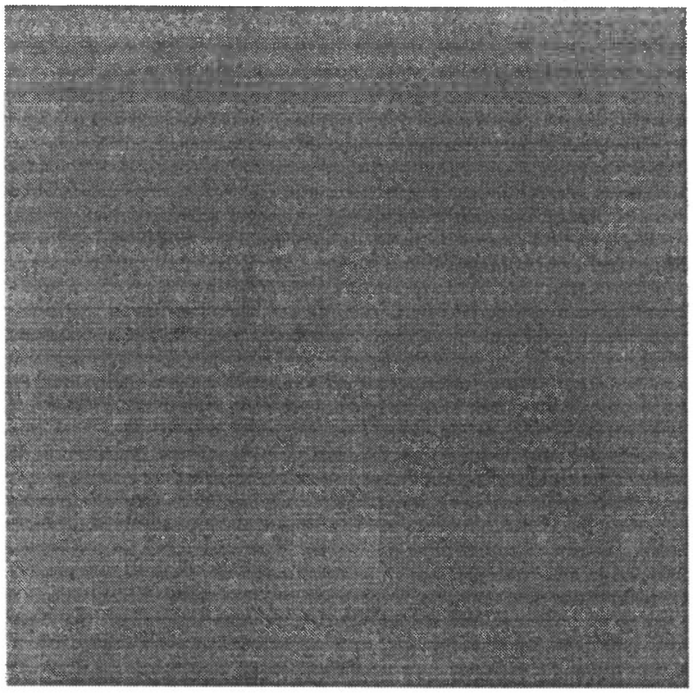
95% Compression (256x256)

RMS = 8.57



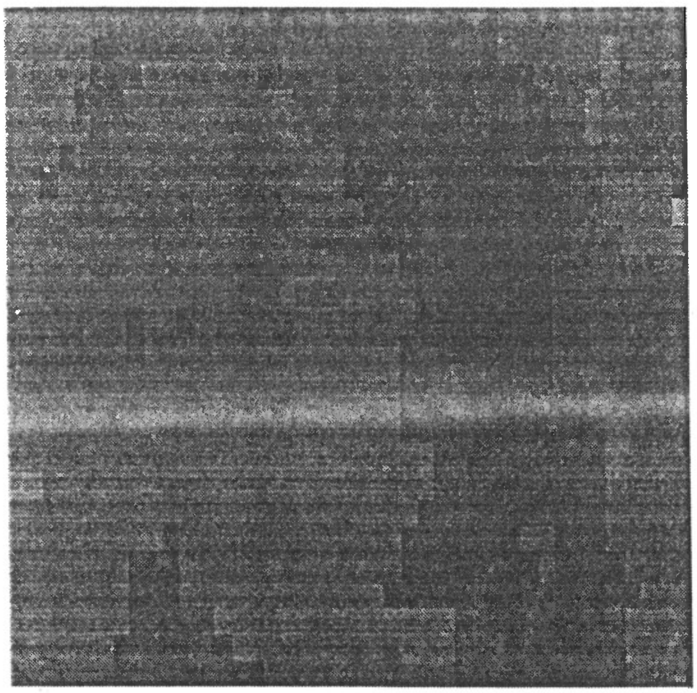
98% Compression (256x256)

RMS = 46.18



99% Compression (256x256)

RMS = 54.29

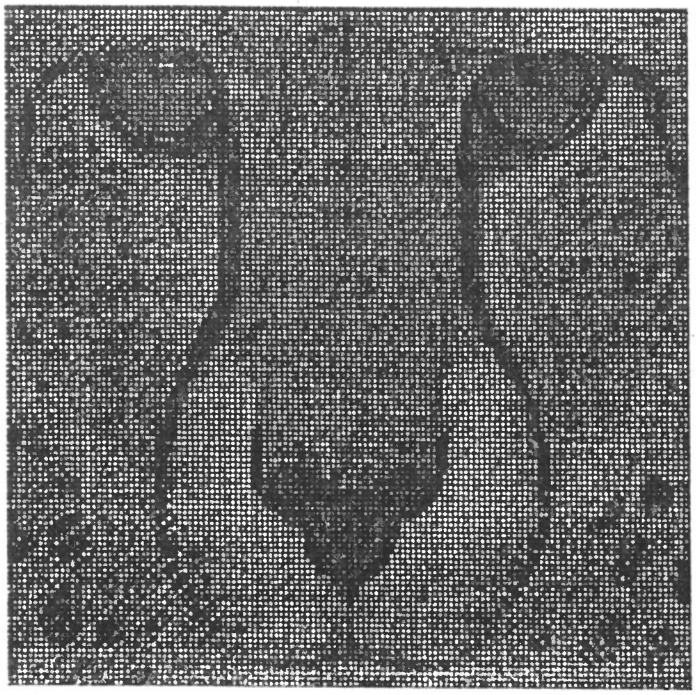


APPENDIX 2

RESULTS FROM THRESHOLDING COMPRESSION

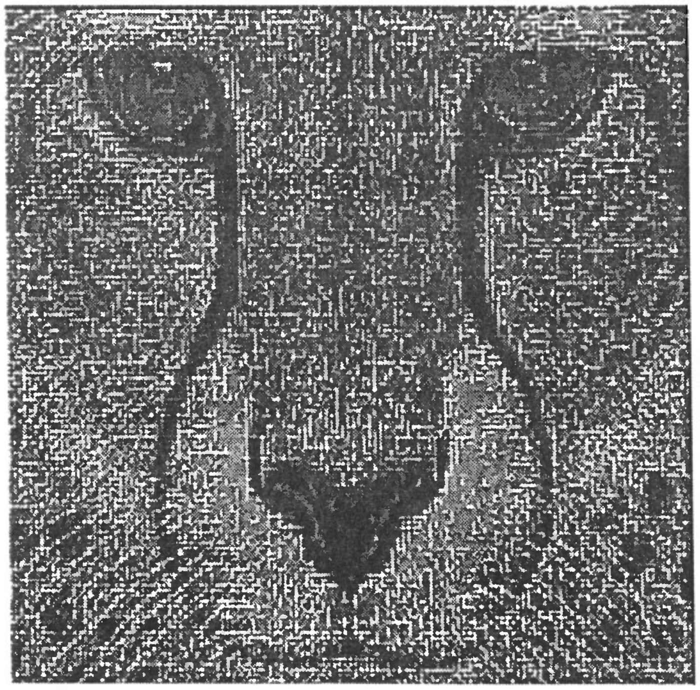
9% Compression (256x256)

RMS = 4.54



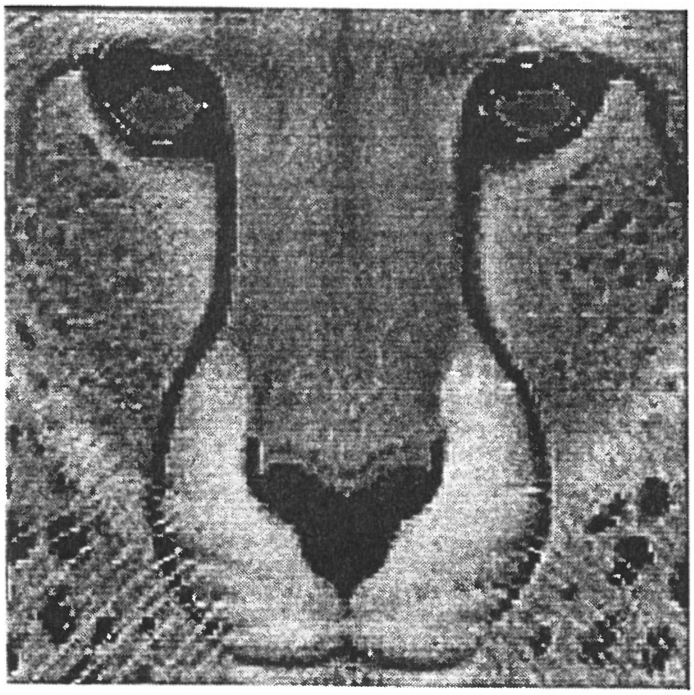
57% Compression (256x256)

RMS = 3.45



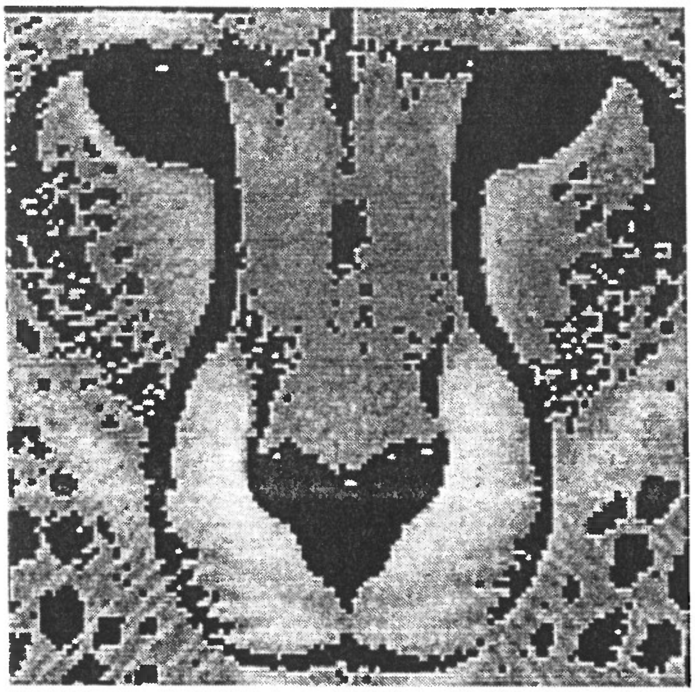
83% Compression (256x256)

RMS = 2.19



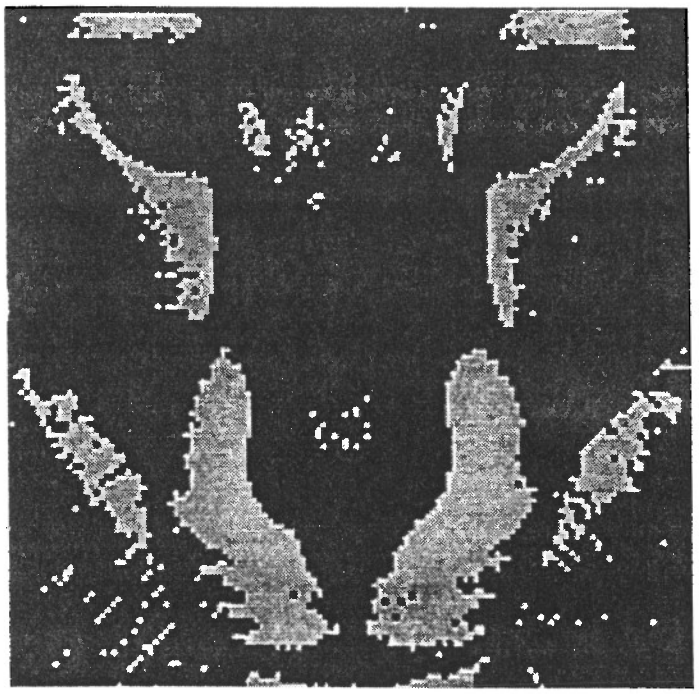
90% Compression (256x256)

RMS = 2.70



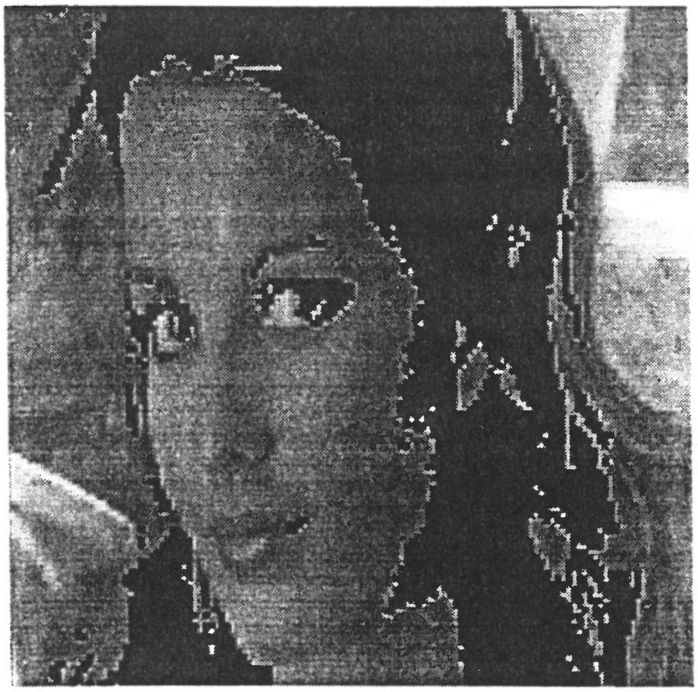
94% Compression (256x256)

RMS = 6.41



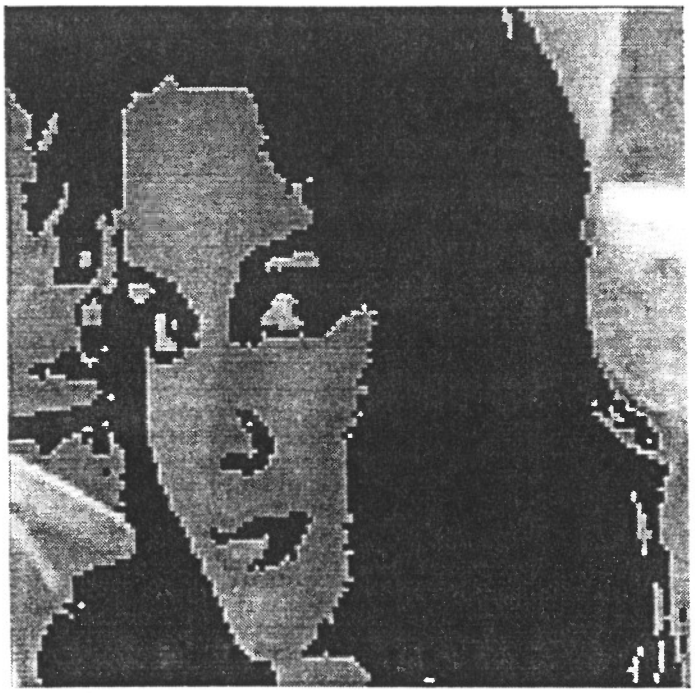
83% Compression (256x256)

RMS = 4.07



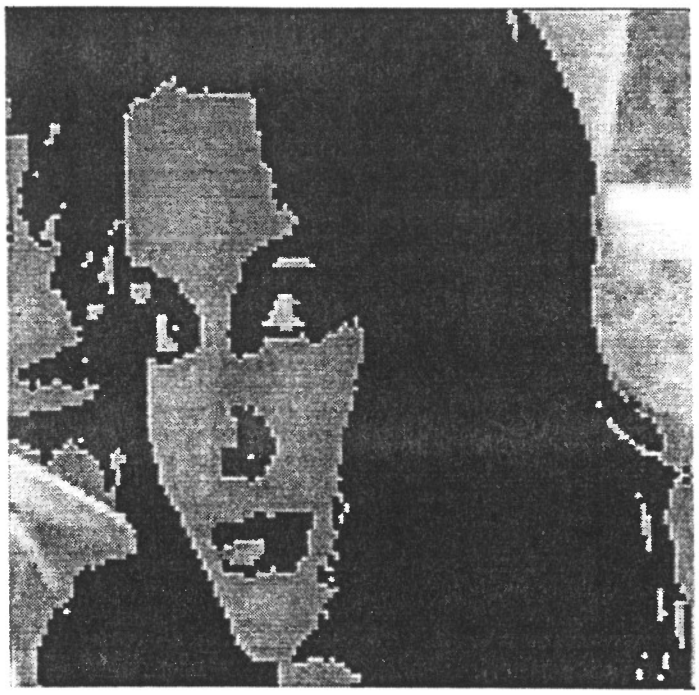
90% Compression (256x256)

RMS = 3.53



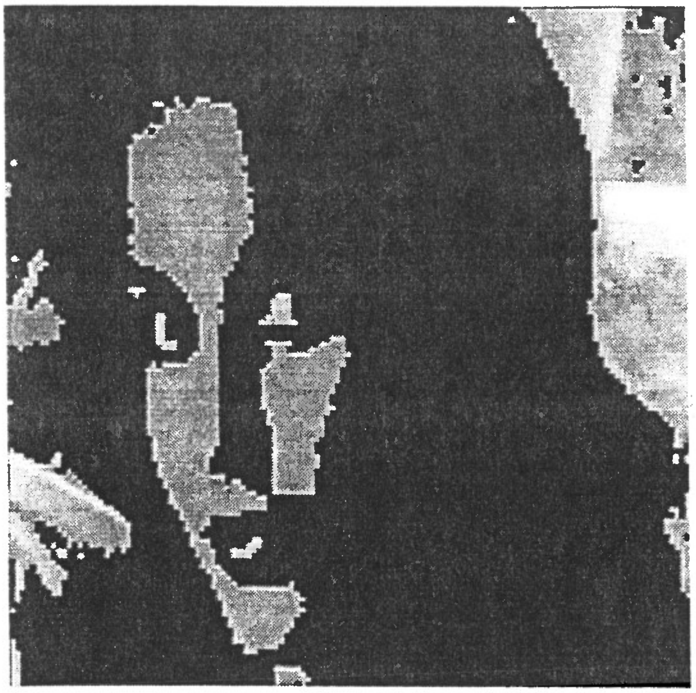
92% Compression (256x256)

RMS = 3.47



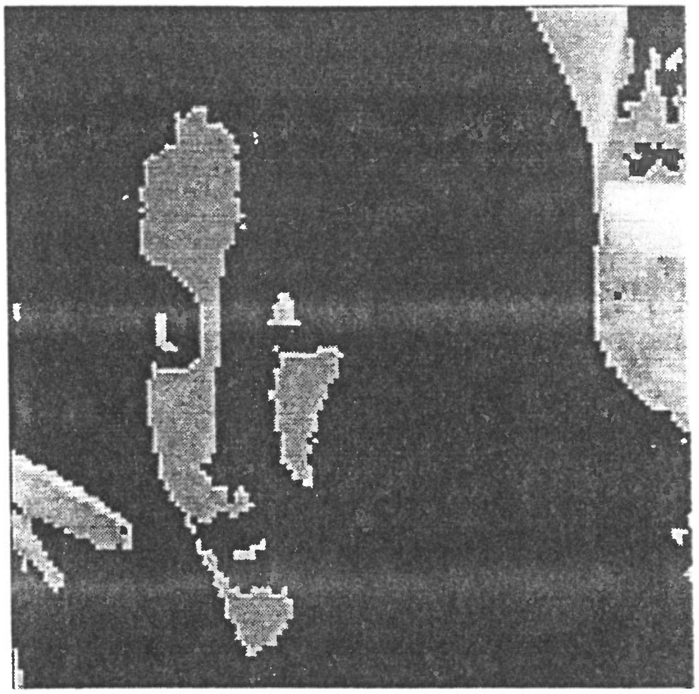
94% Compression (256x256)

RMS = 3.49



95% Compression (256x256)

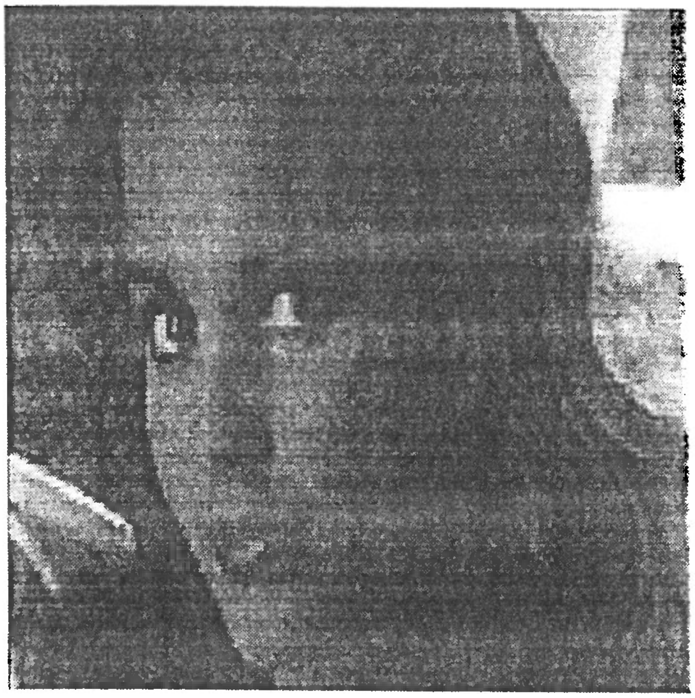
RMS = 3.90



APPENDIX 3
RESULTS FROM ZONE COMPRESSION

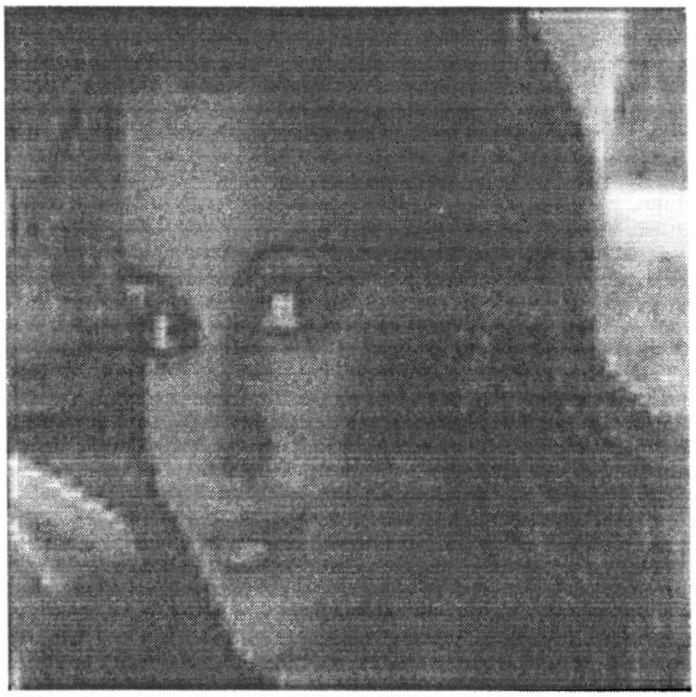
75% Compression (256x256)

RMS = 2.76



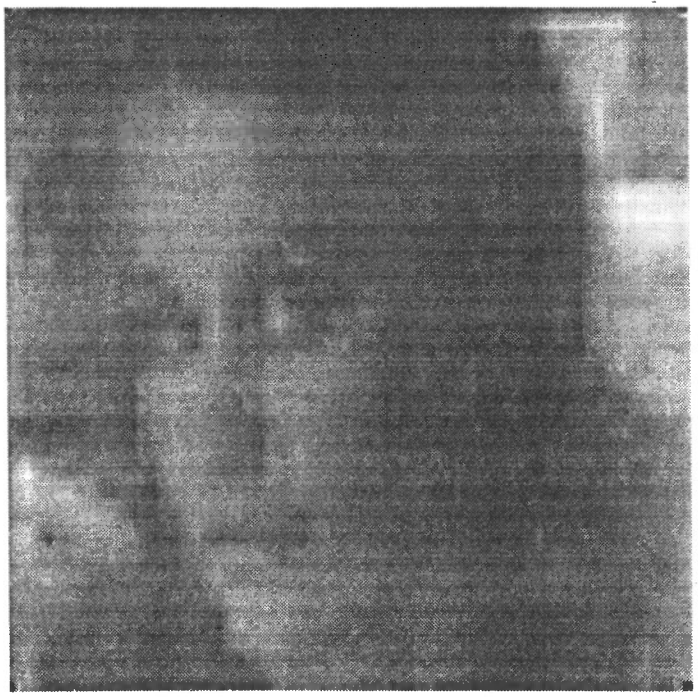
94% Compression (256x256)

RMS = 3.80



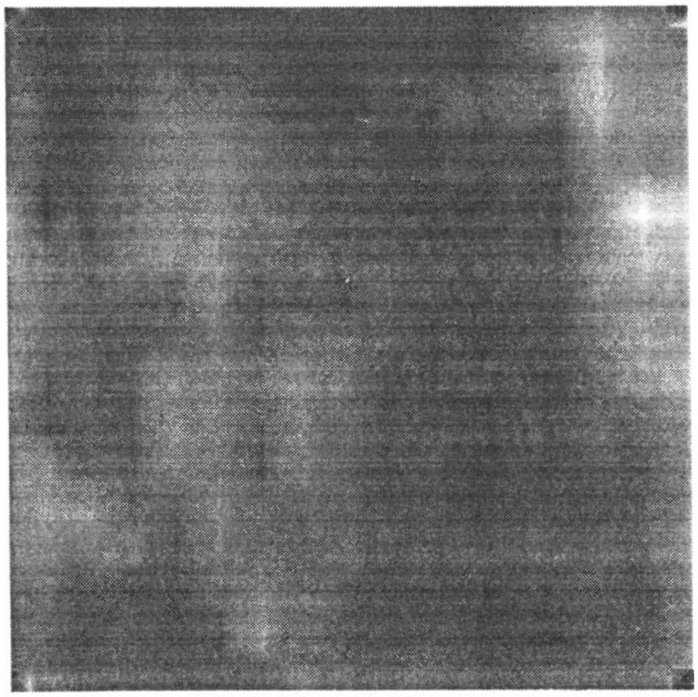
98% Compression (256x256)

RMS = 4.84



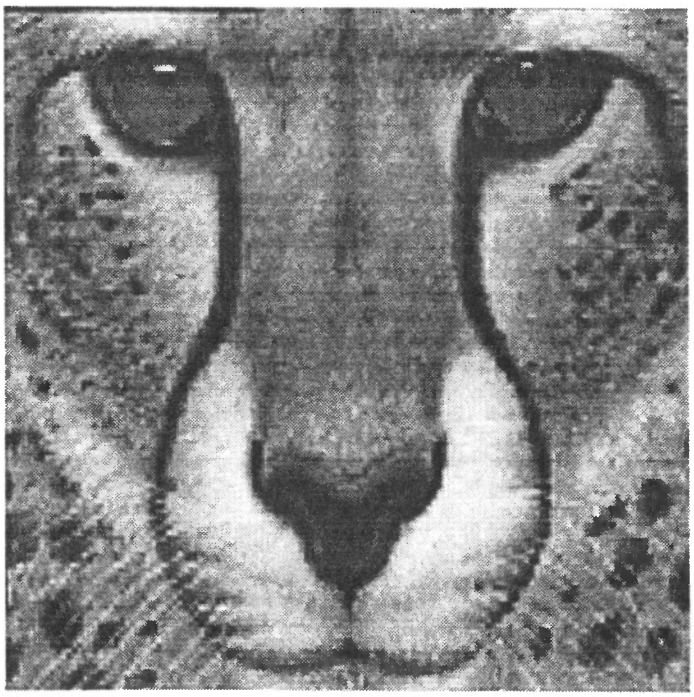
99% Compression (256x256)

RMS = 5.19



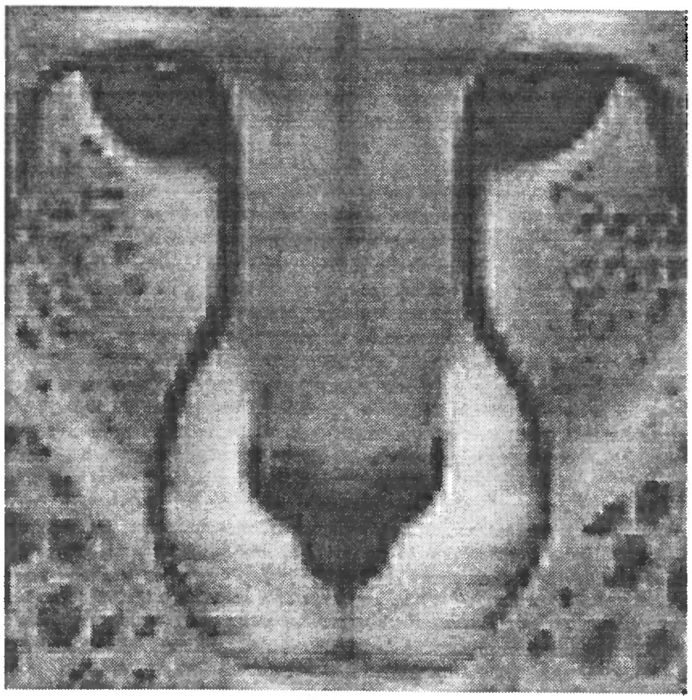
75% Compression (256x256)

RMS = 1.86



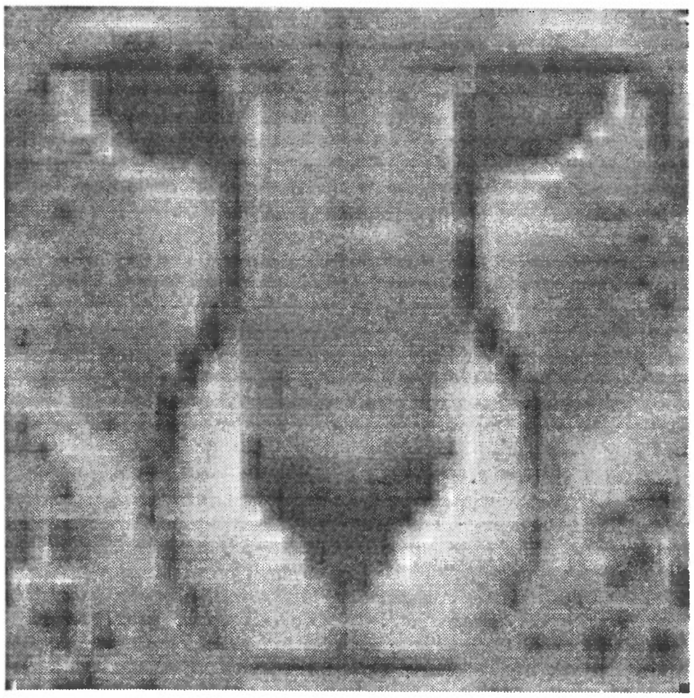
94% Compression (256x256)

RMS = 2.68



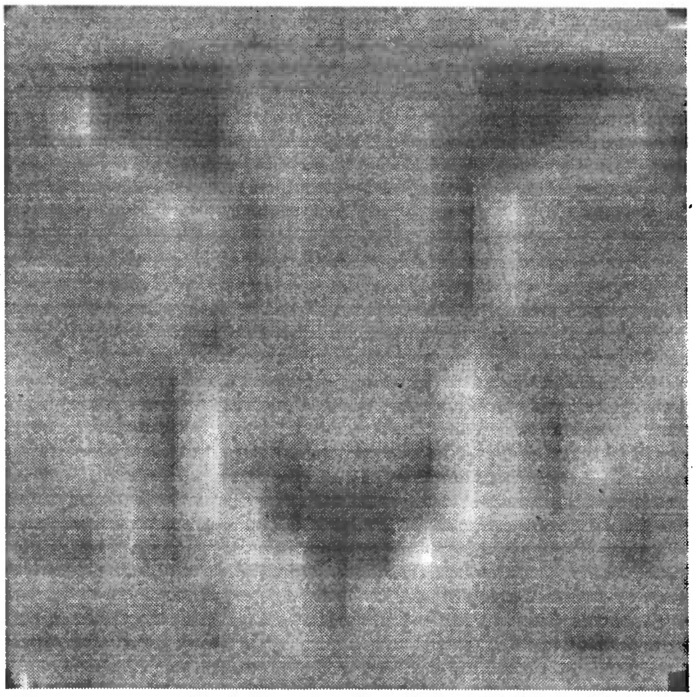
98% Compression (256x256)

RMS = 3.90



99% Compression (256x256)

RMS = 4.79



BIBLIOGRAPHY

1. Cohen, A. and J. Froment, Image Compression and Multiscale Approximation, in Meyer, Y., (ed.) *Wavelets and Applications*, pp. 183-204.

2. Daubechies, I., *Ten Lectures on Wavelets*, Capital City Press, 1992.

3. Daubechies, I., Wavelet Transform: A Method For Time-Frequency Localization, in Haykin, S., (ed.) *Advances in Spectrum Analysis and Array Processing*, pp. 366-416.

4. Fausett, L., *Fundamentals of Neural Networks*, Prentice-Hall, Inc, 1994.

5. Fukushima, K., Cognitron: A Self-organizing Multilayered Neural Network, *Biol. Cybernetics*, vol. 20, pp. 121-136, 1975.

6. Fukushima, K. and S. Miyake, Neocognitron: A New Algorithm For Pattern Recognition Tolerant Of Deformations And Shifts In Position, *Pattern Recognition*, vol. 15, pp. 455-469, 1982.

7. Mallat, S. and S. Zhong, Wavelet Maxima Representation, in Meyer, Y., (ed) *Wavelets and Applications*, pp. 209-216.

8. Mallat, S., Multifrequency Channel Decompositions of Images and Wavelet Models, *IEEE Transactions On Acoustic, Speech, And Signal Processing*, vol. 37, pp. 2091-2111, December 1989.

9. Morris, J. M., V. Akunuri, and H. Xie, More Results on Orthogonal Wavelets with Optimum Time-Frequency Resolution, *SPIE*, vol. 2491, pp. 52-62.

10. Nacken, P., Image Compressiion Using Wavelets, *Wavelets: An Elementary Treatment of Theory and Applications*, pp. 81-91, 1993.

11. Nelson, M. and J. Gailly, *The Data Compression Book*, M&T, New York, 1996.

12. Phillips, D., *Image Processing In C*, R&D Publication, Inc, Lawrence, Kansas, 1994.

13. Press, W. H., S. A. Teukolsky, W. T. Vetterling, and B. P. Flannery, *Numerical Recipes In C*, Cambridge University Press, New York, 1992.

14. Schnorrenberg, F., *Fukushima's Neocognitron: An Implementation*, May 1992.

15. Srikanth, R., R. George, N. Warsi, D. Pubhu, F. E. Petry, B. P. Buckles, A Variable Length Genetic Algorithm for Clustering and Classification, *Pattern Recognition Letters*, pp. 789-800.

16. Strang, G., Wavelets and Dilation Equations, *SIAM Review*, vol. 34, pp. 614-627, December 1989.