New Algorithms for the Classification and Compression of Hyperspectral Images

Bei Xie and Tamal Bose
Electrical and Computer Engineering
Utah State University
Logan, Utah 84322

Erzsébet Merényi
Electrical and Computer Engineering
Rice University
6100 Main Street MS 380
Houston, Texas 77005

***Abstract*— Classification and compression are common operations in image processing. Conventionally, compression and classification algorithms are independent of each other and performed sequentially. In this paper, a new algorithm is developed, where the two operations are combined in order to optimize some given classification metrics. In other words, the compression ratio is maximized under classification constraints. Compression is achieved using Adaptive Differential Pulse Code Modulation (ADPCM), which has an adaptive predictor. The predictor coefficients are updated in real-time by optimizing a cost function based on classification metrics. Optimization is done using a simple genetic algorithm. Computer simulations are performed on hyperspectral images. The results are promising and illustrate the performance of the algorithm under various constraints and compression schemes.**

***Keywords*—GA-ADPCM, SOM, compression, classification**

I. Introduction

Image classification is used to find special features of an image. It is usually performed on the original image, not the image after compression and decompression process. Because of the large storage volume and spectral and spatial sample data redundancy[1], it is better to compress hyperspectral images before classification. Obviously, there will be classification error between pre- and post-compression classifications. To reduce classification error, it is necessary to improve the compression system with feedback from the classifier.

In this paper, we combine the compression system and classification system as in Fig. 1. Classification both on an original image and on a decompressed image is implemented. The error between them is fed back to help design the ADPCM-based compression system. A genetic algorithm based ADPCM is developed to realize this structure. We use the genetic algorithm to choose the best filter coefficients in ADPCM. The fitness function which measures the ADPCM filter performance is the pre- and post-ADPCM classification error. Fig. 2 shows the GA-ADPCM algorithm. In our experiment, two ADPCM predictors are tested to minimize the mean square error in the compression system. One is a Least Mean Square (LMS) predictor, and the other is a Euclidian Direction Search (EDS) predictor. LMS is a very simple algorithm, and a good choice to update the predictor[2]. EDS[3][4] is a relatively new algorithm that aims to increase the convergence speed of the LMS. An unsupervised classification algorithm, self-organizing map (SOM)[5], is used to classify images as SOM has been successfully used to extract useful information from a number of space mission data sets, as well as to perform terrestrial studies[6]. A genetic algorithm (GA) is used to minimize the classification error since GA provides a general approach for searching for global minima or maxima within a bounded, quantized search space[7].

This work was supported in part by NASA AISR grants #NAG5-10716 and NNG06GE95G.

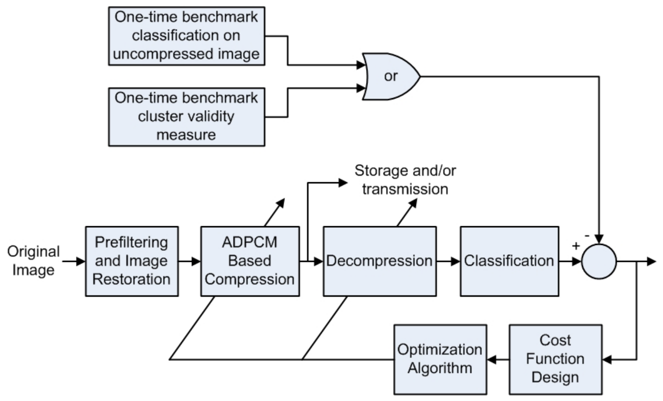
One-time benchmark classification on uncompressed image
One-time benchmark cluster validity measure
or
Storage and/or transmission
Original Image
Prefiltering and Image Restoration
ADPCM Based Compression
Decompression
Classification
+
-
Optimization Algorithm
Cost Function Design

Fig. 1. System combining compression and classification.

II. Review of algorithms

A. REVIEW OF ADPCM SYSTEM[8]

Fig. 3(a) and (b) show the transmitter and receiver of the ADPCM system, respectively. The signal to be transmitted or stored is $s(n)$. The output of the adaptive predictor is $\hat{s}(n)$. The input to the adaptive predictor is $\tilde{s}(n)$. The difference between the actual data and the predicted data is $e(n)$=$s(n)$-$\hat{s}(n)$. The quantized error to be transmitted or stored is $\tilde{e}(n)$. In this paper, an adaptive quantizer named Jayant quantizer[8] is used.

For least-square based algorithms, the exponentially weighted least squares cost function is

$$J_n(\mathbf{w}) \triangleq \sum_{i=1}^{n} \lambda^{n-i} e^2(i), \qquad (1)$$

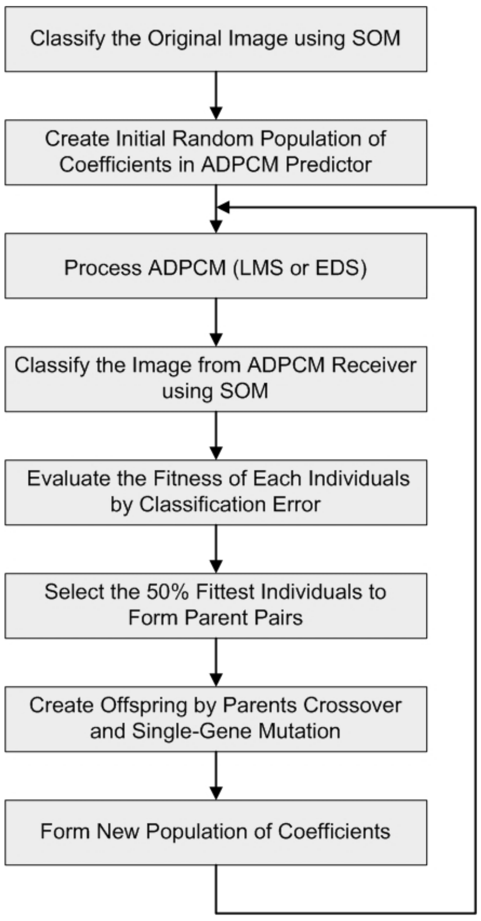
Classify the Original Image using SOM
Create Initial Random Population of Coefficients in ADPCM Predictor
Process ADPCM (LMS or EDS)
Classify the Image from ADPCM Receiver using SOM
Evaluate the Fitness of Each Individuals by Classification Error
Select the 50% Fittest Individuals to Form Parent Pairs
Create Offspring by Parents Crossover and Single-Gene Mutation
Form New Population of Coefficients

Fig. 2. Genetic ADPCM algorithm.

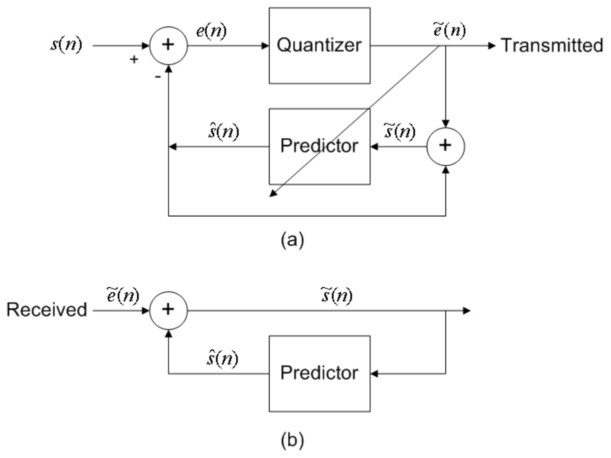

Fig. 3. ADPCM: (a)transmitter,(b)receiver.

where $e(i) = s(i) - \mathbf{w}^T(n)\tilde{\mathbf{s}}(n)$, λ is the forgetting factor, and $s(i), \mathbf{w}(n), \tilde{\mathbf{s}}(n)$ are the desired signal, weight vector, and input signal vector, respectively. Expanding the sum shows that the cost function is quadratic,

$$J_n(\mathbf{w}) = \mathbf{w}^T\mathbf{Q}(n)\mathbf{w} - 2\mathbf{w}^T\mathbf{r}(n) + \sigma_s^2(n), \quad (2)$$

$$\mathbf{Q}(n) = \sum_{i=1}^{n} \lambda^{n-i}\tilde{\mathbf{s}}(i)\tilde{\mathbf{s}}^T(i), \quad (3)$$

$$\mathbf{r}(n) = \sum_{i=1}^{n} \lambda^{n-i}s(i)\tilde{\mathbf{s}}(i), \quad (4)$$

$$\sigma_s^2(n) = \sum_{i=1}^{n} \lambda^{n-i}s^2(i). \quad (5)$$

B. REVIEW OF LMS ALGORITHM

The Least Mean Square (LMS) is a very popular mean square based adaptive algorithm that aims to minimize the squared error in prediction. In image processing, the 2-D LMS algorithm is used. The algorithm is as follows[8]:

```
$Initialization:$
    $\mathbf{w} = \mathbf{0}$
    $\tilde{s}(n_1, n_2) = 0, n_1 \leq 0, n_2 \leq 0$
$Algorithm:$
    $For\ n_2 = 0, 1, 2, ...$
        $For\ n_1 = 0, 1, 2, ...$
        $\tilde{\mathbf{s}}(n_1, n_2) = [\tilde{s}(n_1, n_2)\tilde{s}(n_1, n_2 - 1)...\tilde{s}(n_1, n_2 - M_2)$
        $\tilde{s}(n_1 - 1, n_2)\tilde{s}(n_1 - 1, n_2 - 1)...,$
        $\tilde{s}(n_1 - M_1, n_2 - M_2)]^T$
        $\hat{s}(n_1, n_2) = \mathbf{w}\tilde{\mathbf{s}}(n_1, n_2)$
        $e(n_1, n_2) = s(n_1, n_2) - \hat{s}(n_1, n_2)$
        $\tilde{e}(n_1, n_2) = Quantizer(e(n_1, n_2))$
        $\mathbf{w} = \mathbf{w} + \mu\tilde{\mathbf{s}}(n_1, n_2)\tilde{e}(n_1, n_2)$
        $\tilde{s}(n_1, n_2) = \hat{s}(n_1, n_2) + \tilde{e}(n_1, n_2)$
        $end\ n_1$
    $end\ n_2.$
```

Symbol $\mathbf{w}$ represents the predictor coefficient vector (weights). Here the explicit dependence of $\mathbf{w}$ on the time index n has been dropped. The symbols M_1 and M_2 are the length of coefficients in each dimension. The symbol μ is the step size. Other symbols are the same as in the previous subsection.

C. REVIEW OF EDS ADAPTIVE ALGORITHM

The Euclidean Direction Search (EDS) algorithm is a least-squares based algorithm that attempts to combine the benefits of fast convergence for the Recursive Least Squares (RLS) and the low computational complexity of the LMS. The algorithm is as follows[8],

```
$Initialization:$
    $\mathbf{w} = \mathbf{0}$
    $\tilde{s}(n_1, n_2) = 0, n_1 \leq 0, n_2 \leq 0$
    $\mathbf{Q} = \mathbf{0}$
```

```
  $\mathbf{r} = \mathbf{0}$
$Algorithm:$
  $For\ n_2 = 0, 1, 2, ...$
    $For\ n_1 = 0, 1, 2, ...$
    $\tilde{\mathbf{s}}(n_1, n_2) = [\tilde{s}(n_1, n_2)\tilde{s}(n_1, n_2 - 1)...\tilde{s}(n_1, n_2 - M_2)$
    $\tilde{s}(n_1 - 1, n_2)\tilde{s}(n_1 - 1, n_2 - 1)...,$
    $\tilde{s}(n_1 - M_1, n_2 - M_2)]^T$
    $\hat{s}(n_1, n_2) = \mathbf{w}\tilde{\mathbf{s}}(n_1, n_2)$
    $e(n_1, n_2) = s(n_1, n_2) - \hat{s}(n_1, n_2)$
    $\tilde{e}(n_1, n_2) = Quantizer(e(n_1, n_2))$
    $\tilde{s}(n_1, n_2) = \hat{s}(n_1, n_2) + \tilde{e}(n_1, n_2)$
      $For\ i = 1, 2, ..., N$
      $\epsilon = (\mathbf{q}^i)^T\mathbf{w} - \mathbf{r}_i$
      $a = (\mathbf{q}^i)_i$
      $if\ a \neq 0, \alpha = -\epsilon/a$
      $\mathbf{w}_i = \mathbf{w}_i + \alpha$
      $\mathbf{q}^i = \lambda\mathbf{q}^i + (\tilde{\mathbf{s}}(n_1, n_2))_i\tilde{\mathbf{s}}^T(n_1, n_2)$
      $\mathbf{r}_i = \lambda\mathbf{r}_i + (\tilde{\mathbf{s}}(n_1, n_2))_i s(n_1, n_2)$
      $end\ i$
    $end\ n_1$
  $end\ n_2,$
```

where $\mathbf{Q}(n) = \sum_{i=1}^{n} \lambda^{n-i}\tilde{\mathbf{s}}(i)\tilde{\mathbf{s}}^T(i)$, $\mathbf{q}^i$ is the *i*th vector of $\mathbf{Q}$, $(\mathbf{q}^i)_i$ is the (*i*,*i*)th element of $\mathbf{Q}$, $\mathbf{r}(n) = \sum_{i=1}^{n} \lambda^{n-i}s(i)\tilde{\mathbf{s}}^T(i)$, and $\mathbf{w}_i$, $\mathbf{r}_i$, $(\tilde{\mathbf{s}}(n_1, n_2))_i$ are the *i*th elements of the corresponding vectors. The symbol α is the step size. and $N = M_1M_2$ is the total length of coefficients. Other symbols are the same as in the previous subsection.

D. REVIEW OF SOM ALGORITHM

Self-organizing map (SOM) is an unsupervised neural network-based classification algorithm. The basic idea of SOM is non-linear mapping from the high-dimension input vectors onto a 2-dimension layer of neurons so that the similarities of the input data are reflected by the output neurons[6]. There is a weight between each input vector and corresponding neuron. Neurons compete with each other according to the competition rule, which is usually the shortest Euclidean distance between input vector and weight. Once a neuron wins, its neighbor neurons and the weight are updated[5].

E. REVIEW OF GENETIC ALGORITHM

A genetic algorithm is used to find approximate solutions to optimization and search problems using techniques inspired by evolutionary biology[9]. It first selects an initial population of random individuals. Second, the fitness function of the population is evaluated by using certain criteria. Third, it does some genetic operations including inheritance, mutation, selection, and crossover[9] to the most fit individuals to form the new population. This ensures that more fit individuals have a greater chance to survival and reproduction, while less fit ones are more likely to be discarded[10]. In the next iteration of the algorithm, the new population will be used[9]. This process (Fig. 4) is repeated until the final goal is achieved[11].

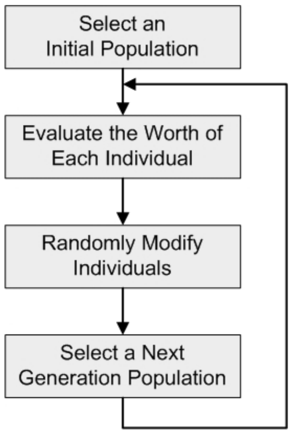
Select an Initial Population
Evaluate the Worth of Each Individual
Randomly Modify Individuals
Select a Next Generation Population

Fig. 4. Genetic algorithm.

In our paper, the individuals are the coefficients in the ADPCM filter. The fitness criterion is a cost function based on the classification error between pre- and post-compression images.

III. GA-ADPCM ALGORITHM

Since the genetic algorithm (GA) has the property of finding the optimal solution and the ADPCM algorithm has the property of minimizing the mean square error, these two algorithms are combined into the GA-ADPCM algorithm that can not only minimize the mean square error but also minimize the classification error between pre- and post-compression images. The flowchart is shown in Fig. 2. First, we generate the initial population of random individuals, which are the different sets of coefficients used in the ADPCM predictor. Second, we implement the ADPCM compression and decompression processing using these different sets of coefficients. Third, we apply SOM to the decompressed images. The application of SOM to the original image should be done at the beginning. Fourth, we calculate the classification error between the decompressed image and the original image. We sort the classification errors of all individuals, and then choose the 50% fittest individuals and apply genetic operations, e.g., crossover and mutation, to them. We create the new coefficients population and do the iteration again.

The fitness function F to be minimized in GA is defined as

$$F = C_e/N, \tag{6}$$

where C_e is the number of pixels classified incorrect, and N is the total pixels of the image. F becomes the percentage of incorrect classified pixels. C_e is obtained by the following steps: First, we do $C_m = C_o - C_g$, where C_o is the matrix containing classification result of the original image, which has all pixels assigned to different classes. C_g is the matrix containing classification result of the image after ADPCM compression and decompression system. C_m is the difference between the two classified images. Second, we assign all the nonzero points in C_m matrix to be 1 and add them together to get the classification error C_e.

The cost function of this algorithm is no longer equation (1) or (2). It is not quadratic, but a more complex function. For instance, it may appear as in Fig. 5. In each part, A, B, and C, respectively, the cost function is quadratic because of the properties of the ADPCM. The LMS and EDS predictors try to find one of the local minima of the cost function, which means trying to find one of the points A, B, and C. However, they may not guarantee the global minimum. Since GA is then used to find the global minimum of the whole cost function, the algorithm GA-ADPCM tries to find point A in this example.

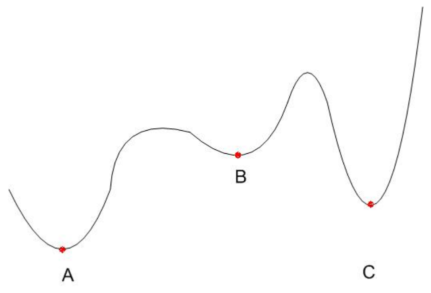

Fig. 5. Cost function of GA-ADPCM.

IV. Simulations

An example of hyperspectral cube is shown in Fig. 6. One frame of the cube is used to be the original image in the simulation and shown in Fig. 7. The size of the original image is 512x512 pixels. To save the storage of the memory on the computer and increase the speed of the performance, the image was divided into blocks. Different block sizes were tried, including 16x16, 32x32, and 64x64 pixels. In SOM, different numbers of classifiers were also tried, including 3 classes, 4 classes, 6 classes, and 8 classes. In all these trials, the performance was similar in that the classification errors were smaller in GA-ADPCM than in ADPCM.

A. *GA-LMS illustration performance*

We first use LMS in the ADPCM system. Fig. 8 is one 64x64 block of the original image, and Fig. 9 is the classification result of the original image. In SOM, 4 classes were used. Fig. 10 is the classification result using LMS only, and Fig. 11 is the classification result using GA-LMS. We can see that all of these classification results are correlated, but small differences are apparent between the images. Our purpose is to reduce the differences, thus reducing classification error. The total classification error using GA-LMS (0.066406) is smaller than that using LMS (0.078125).

In GA, the individuals in each generation have different fitness scores. The most fit individual who has the smallest

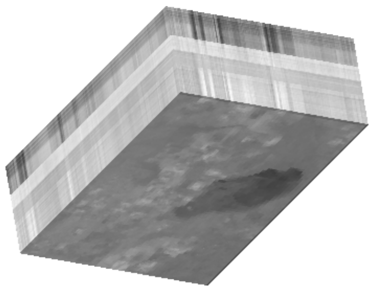

Fig. 6. Hyperspectral cube.

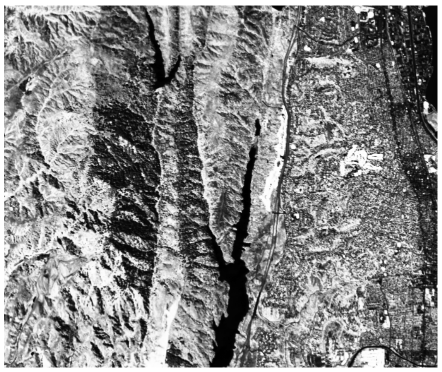

Fig. 7. One frame of the hyperspectral cube.

classification error is saved. Fig. 14 shows the fitness scores of different generations. The red circles represent the smallest fitness in each generation, the blue line represents the largest fitness, and the green marks represent the average fitness scores. We can see that the smallest fitness scores converge to a certain value. The coefficients corresponding to the individual for this fitness score is optimal because they can minimize both the classification error and the mean square error.

B. *GA-EDS illustration performance*

We also use EDS predictor in the ADPCM system. We use the same block of the original image. Fig. 12 is the classification result using only EDS. Fig. 13 is the classification result using GA-EDS. The total classification error using GA-EDS is 0.083008, and the error using EDS is 0.091064. When EDS is used as the predictor, the performance is still improved by using GA-EDS. Fig. 15 shows the fitness scores of different generations in GA-EDS.

C. *Statistic performance of GA-ADPCM*

Table I lists the classification error comparison both between LMS and GA-LMS and between EDS and GA-EDS.

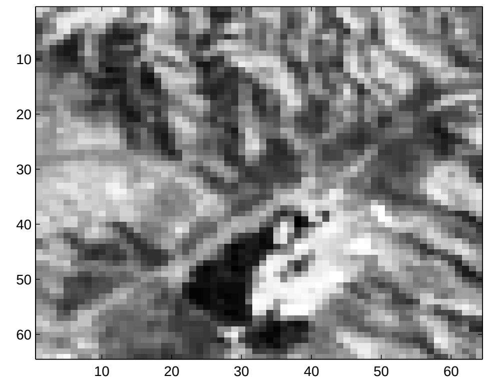

Fig. 8. One block of original image.

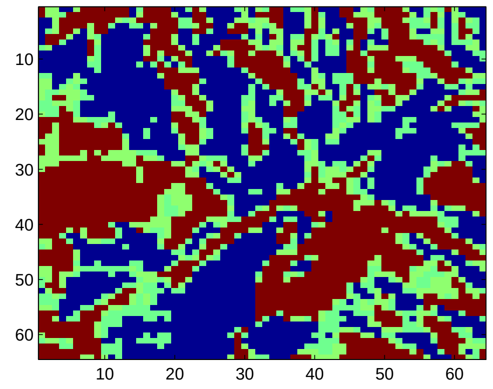

Fig. 11. Classified image from GA-LMS image – colormap color.

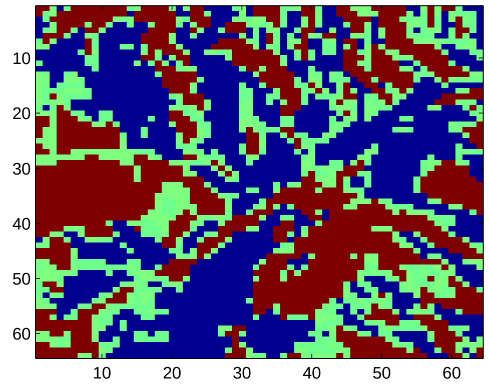

Fig. 9. Classified image from original image – colormap color.

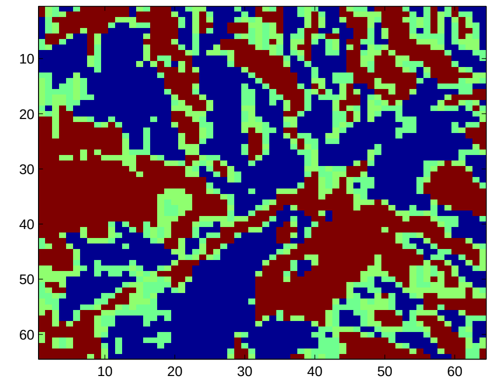

Fig. 12. Classified image from EDS image – colormap color.

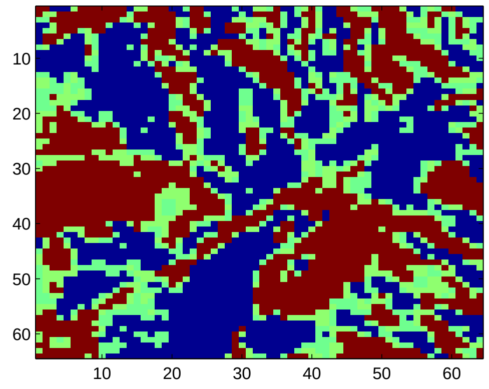

Fig. 10. Classified image from LMS image – colormap color.

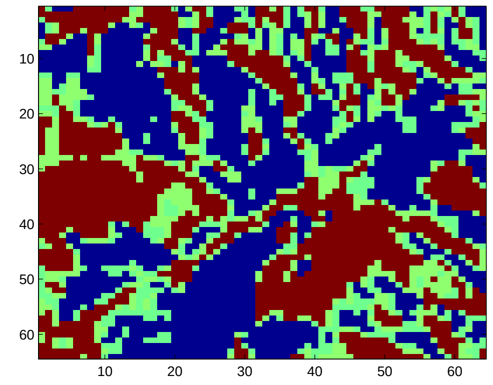

Fig. 13. Classified image from GA-EDS image – colormap color.

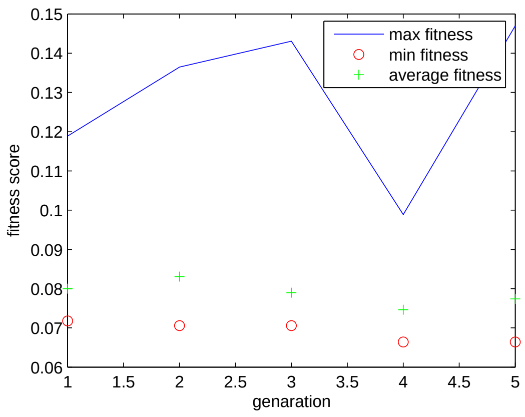

Fig. 14. Fitness score for GA-LMS.

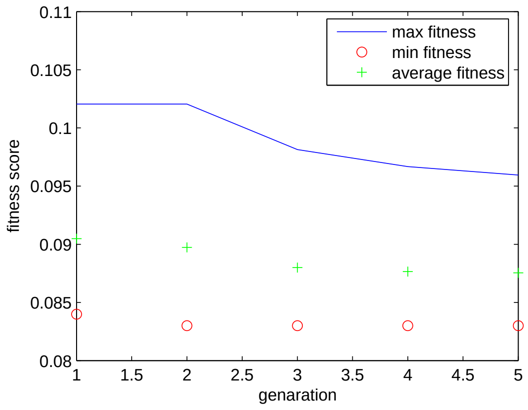

Fig. 15. Fitness score for GA-EDS.

The block size is 64x64 pixels and the class number is 4. The table gives the errors of blocks (1,1) through (1,8). As seen from the table, the GA-ADPCM performs better than the ADPCM in minimizing classification error.

Different block sizes and class sizes were tested. Fig.16 - Fig.23 illustrate the classification-error comparison between the ADPCM and the GA-ADPCM. The red lines represent the error using the ADPCM, while the blue lines are the error using the GA-ADPCM. In these figures, all the red lines are higher than the blue ones, meaning that the classification errors are larger in ADPCM than in GA-ADPCM.

V. Conclusion

We have tried a new structure combining the compression system and the classification system. The corresponding algorithm (GA-ADPCM) is simulated. The filter coefficients in ADPCM are optimized by adding the genetic algorithm. The LMS and EDS predictors minimize the mean squared error. In the genetic algorithm, the fitness function uses the classification error between pre- and post-compression images so that the filter coefficients are also changed to optimize the

TABLE I
Classification error comparison between ADPCM and GA-ADPCM.

Block index	LMS	GA-LMS	EDS	GA-EDS
(1,1)	0.04248	0.034424	0.052734	0.04541
(1,2)	0.060303	0.049561	0.085938	0.070557
(1,3)	0.05127	0.040771	0.070801	0.061279
(1,4)	0.027588	0.023193	0.039551	0.032227
(1,5)	0.04248	0.027588	0.056641	0.04834
(1,6)	0.039063	0.037354	0.05127	0.050781
(1,7)	0.057129	0.049072	0.065186	0.05835
(1,8)	0.069336	0.065674	0.092773	0.080566

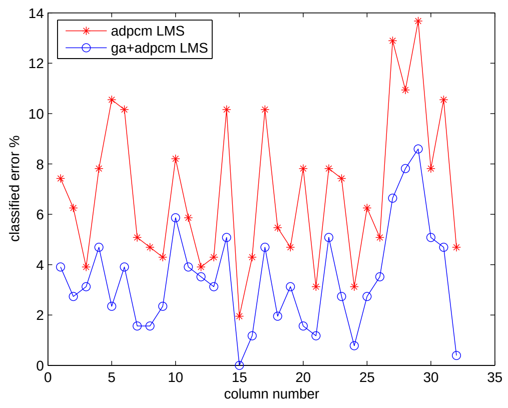

Fig. 16. Classification error comparison between LMS and GA-LMS: block size=16, classes=4.

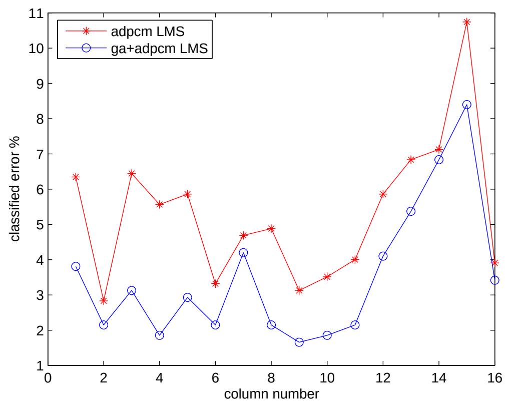

Fig. 17. Classification error comparison between LMS and GA-LMS: block size=32, classes=4.

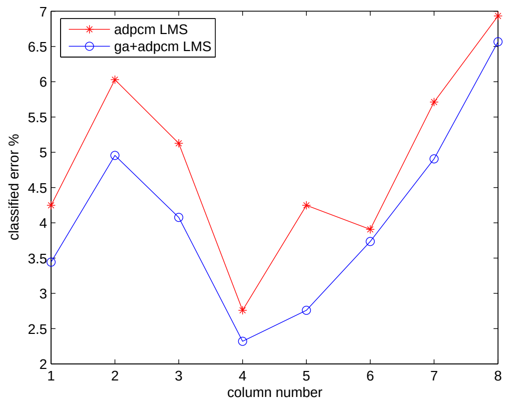

Fig. 18. Classification error comparison between LMS and GA-LMS: block size=64, classes=4.

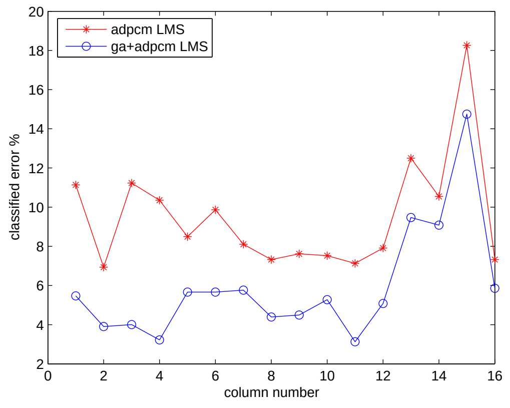

Fig. 21. Classification error comparison between LMS and GA-LMS: block size=32, classes=6.

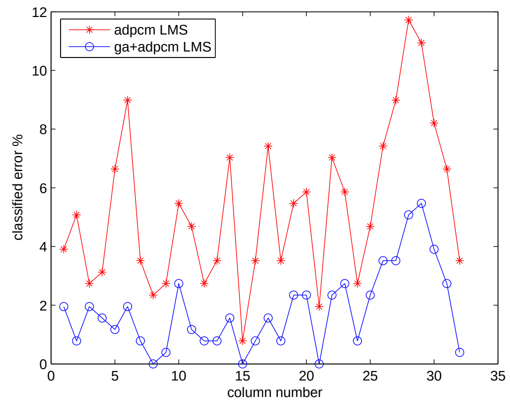

Fig. 19. Classification error comparison between LMS and GA-LMS: block size=16, classes=3.

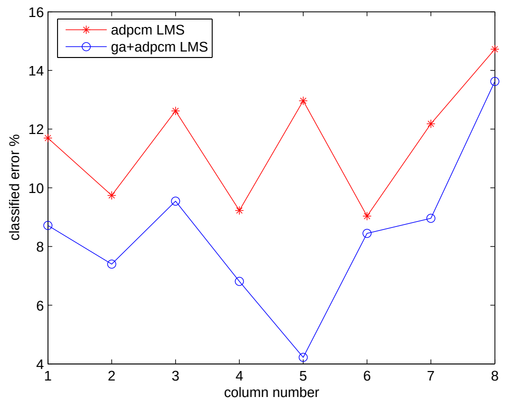

Fig. 22. Classification error comparison between LMS and GA-LMS: block size=64, classes=8.

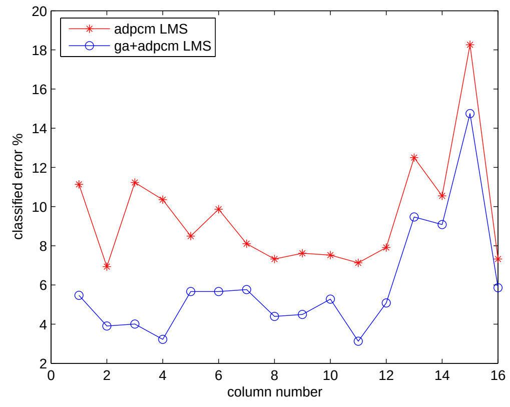

Fig. 20. Classification error comparison between LMS and GA-LMS: block size=16, classes=3.

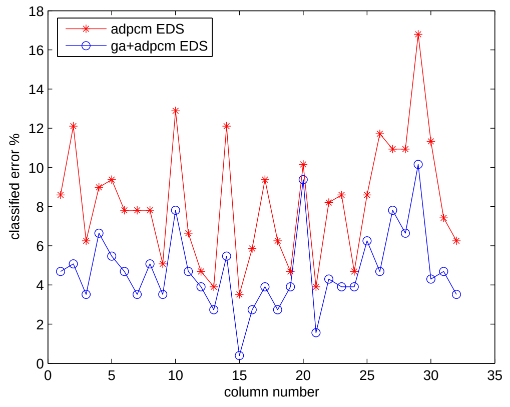

Fig. 23. Classification error comparison between EDS and GA-EDS: block size=16, classes=4.

classification error. The simulation results demonstrate that the algorithm has good performance. However, the cost is the computation complexity. Future work will include combining a multiplier-free adaptive filter[12] with the genetic algorithm. Different classification methods will also be attempted.

REFERENCES

[1] G. Shaw and D. Manolakis, "Signal Processing for Hyperspectral Image Exploitation," *Signal Processing Magazine*, Vo1.19, No.1,January 2002.

[2] M. Larsen, T. Bose, and A. Venkatachalam, "Hyperspectral image restoration and coding," *Asilomar Conference of Signal, Systems, and Computers*, pp. 1740-1744, 2002.

[3] T. Bose and G. F. Xu, "The Euclidean Direction Search Algorithm in Adaptive Filtering," *IEICE Transactions Fundamentals*, vol. E85-A, pp. 532-539, March 2002.

[4] G. F. Xu, T. Bose, W. Kober and J. Thomas, "A fast adaptive algorithm for image restoration," *IEEE Transactions on Circuits and Systems-I*, vol. CAS-46, pp. 216-220, January 1999.

[5] T. Kohonen, *Self-Organizing maps*, Springer, 2nd extended, 1997.

[6] E. Merényi, W. H. Farrand, and P. Tracadas, "Mapping Surface Materials on Mars From Mars Pathfinder Spectral Images With HYPEREYE," *Proc. International Conference on Information Technology (ITCC 2004)*, vol. II, pp. 607 - 614. April 5-7, 2004, Las Vegas, NV, USA.

[7] J.D. Griesbach and D.M. Etter, "Fitness-based exponential probabilities for genetic algorithmsapplied to adaptive IIR filtering," *Asilomar Conference of Signal, Systems, and Computers*, vol. 1, 1998, pp. 523-527.

[8] T. Bose, *Digital Signal and Image Processing*, John Wiley, 2004.

[9] $http://en.wikipedia.org/wiki/Genetic\_algorithms.$

[10] $http://www.cs.bgu.ac.il/\%7Esipper/ga.html.$

[11] S. Ovaska, *Computationally Intelligent Hybrid Systems*, John Wiley, 2005.

[12] T. Bose, A. Venkatachalam, and R. Thamvichai, "Multiplierless adaptive filtering," *Digital Signal Processing*, vol. 12, pp. 107-118, 2002.