http://ansinet.com/itj

ITJ

ISSN 1812-5638

INFORMATION TECHNOLOGY JOURNAL

ANSInet
Asian Network for Scientific Information
308 Lasani Town, Sargodha Road, Faisalabad - Pakistan

Information Technology Journal 6 (3): 457-462, 2007
ISSN 1812-5638

A Tool for Compressing Images Based on Genetic Algorithm

Mohammed A.F. Al-Husainy
Department of Computer Science, Faculty of Sciences and Information Technology,
Al-Zaytoonah University of Jordan P.O. Box 130 Amman (11733) Jordan

Abstract: Genetic Algorithm (GA) has been successfully applied to codebook design for Vector Quantization (VQ). This work uses the facilities of the genetic algorithm (*Crossover* and *Mutation* Operations) to enhance the use of one popular compression method, Vector Quantization (VQ) method. After studying the VQ method, a new algorithm for mixing it with a Genetic Algorithm (GA) is proposed and implemented for testing on some popular image datasets. A good enhancement was recorded for the performance of the (VQ) method when mixed with the (GA). Some fidelity measures are calculated to evaluate the performance of the new proposed algorithm.

Key words: Vector quantization, clustering, crossover, mutation

INTRODUCTION

Vector Quantization (VQ) has been widely used for data compression due to its theoretical advantages compared to scalar quantization schemes. However, the computational complexity of both the codebook design and the vector lookup during the encoding phase is obviously a burden of its realization. Since the codebook can be pre-generated before the encoding process, the efficiency of the vector lookup is comparatively more significant (Chan *et al*., 1994; Hwang *et al*., 2001). Vector Quantization (VQ) has been successfully used in speech and image data compression (Liang *et al*., 1995).

One of the emerging technologies for lossy data (image in this research) compression is Vector Quantization (VQ). Vector Quantization can be used to take advantage of the correlation between neighboring pixels by quantizing pixels in groups (or vectors) rather than individually and symbolically representing the vector with a codeword. The appropriate codeword is chosen from the available codebook by minimizing a given distortion measure. Often, the distortion measure is simply the mean squared error between the quantized pixels and the codevector (Cabral, 1994; Lin and Chang, 2006).

Vector Quantization (VQ) is a source coding methodology with a provable rate-distortion optimality. However, despite more than two decades of intensive research, VQ's theoretical promise is yet to be fully realized in image compression practice (Wu and Wen, 1999). Data compression using Vector Quantization (VQ) has received great attention in the last decade of its promising compression ratio and relatively simple structure. In its simplest implementation, VQ requires breaking the signal to be compressed into vectors (which may be referred to as a blocks). Each vector of the signal to be compressed is compared to the entries of a codebook containing representative vectors. The address of the codebook entry most similar to the signal vector is then transmitted to the receiver, where it is used to fetch the same entry from an identical codebook, thus reconstructing an approximating to the original signal. Compression is obtained because transmitting the address of a codebook entry requires fewer bits than transmitting the vector itself (Midanda-Trigueros *et al*., 1999; Huang *et al*., 1992; Nasrabadi and King, 1988; Kumar, 1999).

A surprising number of everyday problems are difficult to solve by traditional algorithms. A problem may qualify as difficult for a number of different reasons; for example, the data may be too noisy or irregular; the problem may be difficult to model; or it may simply take too long to solve. It's easy to find examples: finding the shortest path connecting a set of cities, dividing a set of different tasks among a group of people to meet a deadline, or fitting a set of various sized boxes into the fewest trucks. In the past, programmers might have crafted a special-purpose program for each problem; now they can reduce their time significantly by using a genetic algorithm (GA) (Al-Rawi and Stephan, 1999; Grant, 1995; Ryu and Eick, 1995; Louis, 1997; Ou and Chen, 2006).

Genetic Algorithm (GA) has been successfully applied to codebook design for Vector Quantization (VQ). However, most conventional GA-based codebook design methods need long runtime because candidate solutions must be fine tuned by LBG. Codebook design is the key problem of VQ and the generated codebook has more effect on the compression performance. The traditional codebook design method*LBG algorithm is affected by the initial codebook, often generates the local optimal codebook and needs intensive computation. Research

efforts in codebook design have been concentrated in two directions: to generate a better codebook that approaches the global optimal solution and to reduce the computational complexity (Huang *et al*., 2001).

VQ using GA (GAVQ): A proposed algorithm: In this section, a new algorithm for Image compression is suggested. This algorithm tries to exploit the facilities of Genetic Algorithm (GA) and then uses these facilities to enhance the use of the Vector Quantization method. The main steps of the proposed algorithm can be stated as follows:

Step 1: Problem Representation (focusing on the choice of a suitable representation of the problem).
Step 2: Clustering (grouping the most similar input instances in sets that have common characteristics between these input instances).
Step 3: Genetic Operations (performing the Crossover and Mutation operations on the elements of the sets that are produced from Step 2).
Step 4: Merging (merging those sets becomes nearest after performing the genetic operations in Step3).
Step 5: Performance Evaluation and Termination Criteria (testing the performance of the algorithm at each generation and termination criteria, if the termination criteria are satisfied then STOP, otherwise GOTO Step 2).

Problem representation: The first step of the GAVQ algorithm is the preparing step, which involves the representation of the problem in such a way that it becomes suitable to be applied by the GAVQ algorithm.

Vector representation: Initially, the image is divided into a non-overlapped blocks of dimension H×W (for examples 2×2, …). Each block is then represented as a vector of D-dimension (i.e., D = H×W). Now, consider the case that one has to represent a given D-dimensional input vector $\vec{X}_i = (x_1, x_2, x_3, ..., x_D)$ with a particular codeword $\vec{C}_j = (c_1, c_2, c_3, ..., c_D)$ selected as the best representative of the vector $\vec{X}_i$ within a codebook (i.e., the codevectors are extracted from the input vectors). The selection is based on the minimum Euclidean distance criterion. The VQ approach is the process that finds a mapping from the set of N input vectors $(\vec{X}_1, \vec{X}_2, \vec{X}_3, ..., \vec{X}_N)$ to the M output vectors (codewords or codevectors) $(\vec{C}_1, \vec{C}_2, \vec{C}_3, ..., \vec{C}_M)$ these M codevectors represent the codebook as shown in Fig. 1.

Chromosomal representation: For any GA, a chromosome representation is needed to describe each

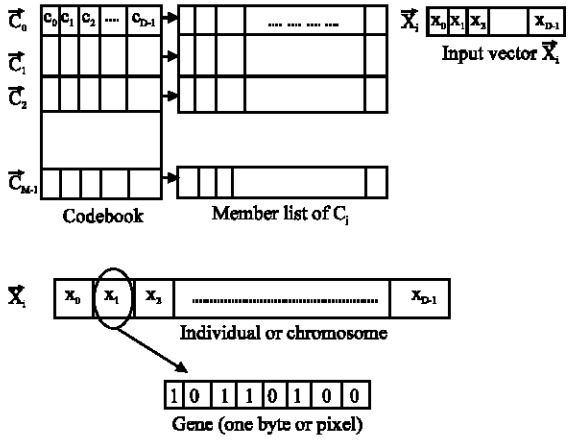

Fig. 1: Problem representation

individual in the population. The representation scheme determines how the problem is structured in the GA and also determines the genetic operators that are used. Each individual or chromosome is made up of a sequence of genes from a certain alphabet (i.e., letters, integer numbers, floating point number or bytes). Consider an input vector $\vec{X}_i = (x_1, x_2, x_3, ..., x_D)$ be an individual (chromosome) in the population that consists of N individuals (input vectors), each chromosome consists of D number of bytes (genes), so the chromosome contains a string of (D*8) binary digits or bits (i.e., 0 or 1), where (D*8) represents a chromosome length, as shown in Fig. 1.

Genetic algorithms use a number of parameters to control their evolutionary search for the solution to their given problem. These parameters include selection operator, crossover operator, mutation operator, crossover rate, mutation rate and maximum number of generations.

At the start of the proposed algorithm, the population size is equal to N, which contains all input vectors $(\vec{X}_1, \vec{X}_2, \vec{X}_3, ..., \vec{X}_N)$. Also, the codebook itself contains all input vectors as the codevectors $(\vec{C}_1, \vec{C}_2, \vec{C}_3, ..., \vec{C}_M)$ initially (i.e., M = N).

Clustering: This operation assigns each vector $\vec{X}_i$ in the population to the codevector $\vec{C}_j$ (for j = 0,1,2, …, M-1) according to the similarities between $\vec{X}_i$ and $\vec{C}_j$, in another words, a vector $\vec{X}_i$ is assigned to the codevector $\vec{C}_j$ if $\vec{C}_j$ is the nearest codevector to $\vec{X}_i$, such that:

$$D(\vec{X}_i, \vec{C}_j) \le D(\vec{X}_i, \vec{C}_l)(j, l=0,1...M-1),(i=0,1...N-1), j \ne 1 \quad (1)$$

where $D(\vec{X}_i, \vec{C}_j)$ means the distortion measure between two vectors of D-dimension given as:

$$D(\vec{X}_i, \vec{C}_j) = \sum_{k=0}^{D-1} |x_k - c_k| \quad (2)$$

The selection of the nearest codevector needs search through all the individuals in the population using a full sequential search. This may take a long time (for a large number of input vectors) at the first generation, but it becomes suitable more and more after some generations.

The clustering operation produces to the next step (i.e., Step 3) a codebook of M codevectors $(\vec{C}_1, \vec{C}_2, \vec{C}_3, ..., \vec{C}_M)$ where M≤N. Each codevector $\vec{C}_j$ involves a list of indexes (members) that represent all input vectors (members) which belong to this codevector.

Genetic operations: Genetic operations (crossover and mutation) are applied to each element $\vec{C}_j$ of the codebook to get a more suitable codevector that represents its members.

Initially, before the start of the algorithm, the parameters that control the working of the GA must be set. The random selection of individuals (parent(s)) for genetic operations is considered to be the selection operator. The crossover operator is a single point (byte) in each individual (parent). The complement method is considered to be the mutation operator. Each crossover and mutation operations are applied on 50% of individuals in the population for each generation. The maximum number of generations is considered 20.

Crossover operation: For each codevector $\vec{C}_j$ in the codebook, select randomly from its member's list one index for example I (i.e., one input vector $\vec{X}_i$) as a first parent and assume that the codevector $\vec{C}_j$ itself is to be the second parent. In the next step, one point (byte) in each parent is randomly selected (x_k and c_k for k = 0,1,2,…,D-1) and then exchange these points between the two parents to produce two child (i.e., two new vectors) $\vec{V}_1$ and $\vec{V}_2$.

After that, the checking step begins to check the most adequate vector among the vectors $(\vec{C}_j, \vec{V}_1 \text{ and } \vec{V}_2)$ to replace it as a new codevector $\vec{C}_j$ that represents its member's list as a better codevector and ignore the two others. This check process is performed by calculating the total distortion over the members (vectors) that belong to the codevector $\vec{C}_j$, for each one of the three vectors (i.e., TD($\vec{C}_j$), TD($\vec{V}_1$) and TD($\vec{V}_2$)). The Total Distortion (TD) can be calculated as follow, note that y_j in Eq. (3) represents the jth byte (gene) in each of the vectors ($\vec{C}_j$ or $\vec{V}_1$ or $\vec{V}_2$) and x_i represents the ith input vector (member) that belong to the codevector $\vec{C}_j$:

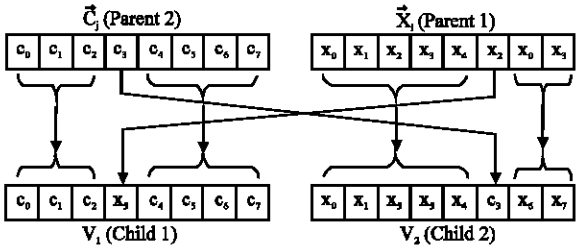

Fig. 2: Crossover operation

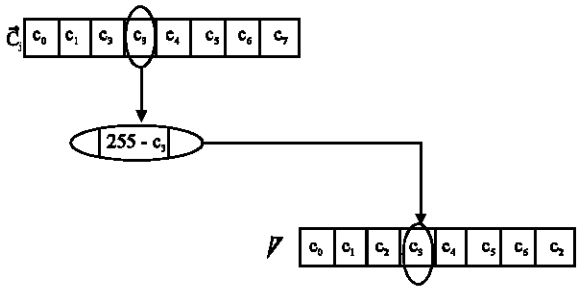

Fig. 3: Mutation operation

$$TD(\vec{Y}) = \sum_{i=0}^{E-1} \sum_{j=0}^{D-1} |y_j - x_{ij}| \quad (3)$$

where E is the number of members (vectors) in the list of members of the codevector $\vec{C}_j$ and D is the vector dimension (i.e., chromosome length in byte). The crossover operation is shown in Fig. 2.

Mutation operation: For each codevector $\vec{C}_j$ in the codebook, select randomly one point (gene) and complement it. Then, replace the new gene in the place of the old one to produce a new vector $\vec{V}$. Now, calculate the total distortion for the new vector TD($\vec{V}$) as in Eq. (3) and compare it with the total distortion TD($\vec{C}_j$) for the original codevector. The new vector $\vec{V}$ is set as a new codevector in place of the original codevector $\vec{C}_j$ if TD($\vec{V}$) ≤ TD($\vec{C}_j$), otherwise ignore this new vector $\vec{V}$. The mutation operation is shown in Fig. 3.

Merging: The goal of this step is to merge each two codevectors $\vec{C}_i$ and $\vec{C}_j$ if the rate of the distortion per each two genes within the two codevectors (D($\vec{C}_i$, $\vec{C}_j$)/ D) ≤ ε, where ε is a small number less than one (0.05 in this algorithm), D is the vector dimension and I, j = 0,1,2, …, M-1 (where M is the number of codevectors in the codebook).

Performance evaluation and termination criteria: After each generation, some performance evaluation and

termination measures are calculated to decide if the algorithm goes to do a next generation or to stop. Signal to Noise Ratio (SNR), Peak Signal to Noise Ratio (PSNR) and Normalized Mean Squared Error (NMSE) should be calculated between each input vector and the reproduction codevector within the codebook that these input vectors belong to as shown in Eq. 4-6. This means really that these measures are calculated between each pixel in the source image file and the reconstructed image file after the de-compression operation is done. For simplicity, let b(I, j) and $\hat{b}$ (I, j) be the pixel in the raw I and column j within the source and decompression image file, respectively.

$$SNR_{db} = 10 \log_{10}\left[\frac{\sum_{i=0}^{N-1}\sum_{j=0}^{N-1}\hat{b}^2(i,j)}{\sum_{i=0}^{N-1}\sum_{j=0}^{N-1}\left[\hat{b}(i,j)-b(i,j)\right]^2}\right] \quad (4)$$

$$PSNR_{db} = 10 \log_{10}\left[\frac{\left[\text{peak value of } b(i,j)\right]^2}{\sum_{i=0}^{N-1}\sum_{j=0}^{N-1}\left[\hat{b}(i,j)-b(i,j)\right]^2 / N^2}\right] \quad (5)$$

$$NMSE = \frac{\sum_{i=0}^{N-1}\sum_{j=0}^{N-1}\left[b(i,j)-\hat{b}(i,j)\right]^2}{\sum_{i=0}^{N-1}\sum_{j=0}^{N-1}\left[b^2(i,j)\right]^2} \times 100 \quad (6)$$

The peak value of b(I, j) is the total dynamic range of the input image and N is the source image dimension.

To check the compression performance, the values of Compression Ratio (CR) and Bit Per Pixel Rate (BR) are calculated. The compression ratio is the amount of compression, while the BR rate is the number of bits required to represent each pixel value of the compressed image. They are calculated in two forms, On-line (when the codebook is taken into consideration) and Off-line (when the codebook is not taken into consideration):

On-line:

$$BR = \frac{\text{VQ file Size in bits + codebook size in bits}}{\text{Source image file size in pixel}} \quad (7)$$

$$CR = \frac{\text{Source image file size}}{\text{VQ file size + codebook size}} \quad (8)$$

Off-line:

$$BR = \frac{\text{VQ file size in bits}}{\text{Source image file size in pixel}} \quad (9)$$

$$CR = \frac{\text{Source image file size}}{\text{VQ file size}} \quad (10)$$

The better compression performance of the algorithm is with the highest compression ratio (the least BR) and the highest PSNR.

The termination criterion in this algorithm is that the algorithm will stop on at least if one of the following condition is satisfied:

- SNR ≥ 22.0 and CR ≥ 4.0 or:
- The number of generation reached the Max, (where Max is the maximum number of generations that are allowed in the algorithm (Max = 20 in this algorithm)).

EXPERIMENTAL RESULTS

The Table 1 show the results that are obtained after applying the (GAVQ) proposed algorithm and one of the classical VQ algorithms (k-means in this Table) on a set of 256 gray-scale levels image data files (Image 1-4 of size 512×512 pixels; Fig. 4). All programs are written by using Visual C++ language (version 6.0) and these programs are executed on the Pentium III (500 MHZ) personal computer.

From the comparison of Table 1, we can note :

- The GAVQ algorithm success to increase CR with saving the SNR in the acceptable range.
- The number of codevector is less than the classical k-means algorithm.
- The crossover and the mutation operations in the above table represent the successful crossover and mutation operations that are participate to increase the CR. Because the algorithm excludes any crossover and mutation operations that are not causing to increase the CR.
- Inspite of the major problem (i.e., the long time) of using GA in any application. The required time for compression when using the GA (in this work), by mixing it with the VQ method, is near to the required time by the k-means. Certainly, it is not suitable to use this algorithm in the real-time systems because the genetic algorithm required long time (especially when the population become very large).

Table 1: Comparison table between GAVQ and classical k-means VQ compression algorithm

File	Compression algorithm	SNR (db)	PSNR (db)	On-line CR (x:1)	Vector Dimension 4 byte: No. of code vectors	No. of input vectors	No. of crossover operations	No. of mutation operations	No. of generations	Time of compression operation (second)
Image 1	GAVQ	24.12	33.25	3.05	1013	65536	911	1044	10	46.6
	VQ (k-means)	23.84	32.65	2.69	1788	65536	-	-	-	15.9
Image 2	GAVQ	24.71	31.45	3.08	822	65536	944	1038	7	36.16
	VQ (k-means)	24.55	31.29	2.72	1570	65536	-	-	-	15.1
Image 3	GAVQ	25.54	31.75	3.96	151	65536	266	273	4	14.4
	VQ (k-means)	24.64	30.89	3.12	552	65536	-	-	-	5.3
Image 4	GAVQ	27.64	30.81	3.12	521	65536	704	726	8	32.23
	VQ (k-means)	27.21	30.44	3.09	743	65536	-	-	-	11.31

Image 1 Image 2 Image 3 Image 4

Fig. 4: Image data files

- Table 1 presented some examples of image files of a small size. Certainly, when using a file of large size, the algorithm firstly divide the file into a number of segments and then apply the steps of the algorithm on these segments to reduce the amount of time and memory space that are required to complete the compression operation.
- Obviously, when we used the SNR and PSNR measures to calculate the distortion (noise) that will appear in the image data during the compression operation is not enough to ensure that this noise not causes a bad effects in the view of the image. We can ensure the effect of this noise by adding another subjective test to the algorithm, but for comparing the performance of the proposed algorithm and another classical algorithm, it is obvious to use objective test as SNR and PSNR.

CONCLUSIONS

Vector Quantization (VQ) compression method was first studied and implemented, then the new method GAVQ that makes use of the GA to design the VQ codebook was proposed, it exploits the crossover and the mutation operations of the genetic algorithm. The GAVQ algorithm was tested on some image files. The recorded results showed that the idea of combining the genetic algorithm and the vector quantization compression method supports well the use of the (VQ) method.

REFERENCES

Al-Rawi, H. and J.J. Stephan, 1999. Genetic Algorithm Based Image Segmentation. Proc. of CATAEE'99, Philadelphia University, Jordan, pp: 78-82.

Cabral, J., 1994. 3D Vector Quantization of Magnetic Resonance Images. Internet Paper, http://www.data-compression/vq.html.

Chan, Y.H., W.C. Siu and K.M. Lam, 1994. A Novel VQ Encoding Algorithm Based on Adaptive Searching Sequence. IEEE Int. Symposium on Speech, Image Processing and Neural Networks, Hong Kong, pp: 164-167.

Grant, K., 1995.An Introduction to Genetic Algorithms. C/C++ Users J., pp: 45-58.

Huang, C.M., Q. Bi and G.S. Stiles, 1992. Fast full search equivalent encoding algorithms for image compression using vector quantization. IEEE Trans. Image Proc., 1: 413-416.

Huang, H.C., J.S. Pan, Z.M. Lu, S.H. Sun and H.M. Hang, 2001. Vector quantization based on genetic simulated annealing. Signal Processing, 81: 1513-1523.

Hwang, W.J., C.F. Chine and S.L. Hong, 2001. Genetic fuzzy entropy-constrained vector quantization. J. Chinese Inst. Eng., 24: 369-377.

Kumar, R., 1999. Codebook Design for Vector Quantization Using Multiobjective Genetic Algorithms. Internet Paper, http://www.rdg.ac.uk/~ssr97jdk/MPSN/Kumar1codebookMPSN.ps.gz.

Liang, K.M., C.M. Huang and R.W. Harris, 1995. Compression between adaptive search and bit allocation algorithms for image compression using vector quantizetion. IEEE Trans. Image Proc., 4: 1020-1023.

Louis, S.J., 1997. Genetic Algorithm and Design. Internet Paper, http://www.cs.unr.edu/~sushil/papers/thesis/thesishtml/node2.html.

Lin, C.Y. and C.C. Chang, 2006. Hiding data in VQ-compressed images using dissimilar pairs. J. Comput., 17: 3-10.

Midanda-Trigueros, A., J.M. Val-Bueno and A.R. Figueiras-Vidal, 1999. The Multiple Representation Problem in Genetic Approaches for Index Assignment in Vector Quantization Codebook Design. Internet Paper, http://www.ehis.nary.mil/tp/humanscience/papers/art18.pdf.

Nasrabadi, N.M. and R.A. King, 1988. Image coding using vector quantization: A review. IEEE Trans. Commun., 36: 957-969.

Ou, C.M. and J.J. Chen, 2006. Hybrid genetic algorithm for design of robust communication systems. J. Software, 1: 24-31.

Ryu, T.W. and C.F. Eick, 1995. MASSON: Discovering Commonalties in Collection of Objects Using Genetic Programming. Internet Paper, URL: http://www.cs.uh.edu/~twryu.

What is Genetic Programming, 1999. Internet Paper, www.genetic-programming.org.

Wu, X. and J. Wen, 1999. Conditional entropy coding of VQ indexes for image compression. IEEE Trans. Image Proc., 8: 1005-1013.