Vector Quantization Based on Self-Adaptive Particle Swarm Optimization

Manish Kumar1 *, Rajiv Kapoor2, Tarun Goel3
1Department of Electronics & Comm. Engg., Amity School of Engg. and Tech., New Delhi
2Department Electronics & Comm. Engg., Delhi Technological University, New Delhi
3Department of Instru. & Control Engg., Amity School of Engg. and Tech., New Delhi
(Received 22 November 2009, accepted 12 January 2010)

Abstract: This article presents a fuzzy self-adaptive particle swarm optimization (FSAPSO) learning algorithm to extract a near optimum codebook of vector quantization (VQ) for carrying on image compression. The fuzzy self-adaptive particle swarm optimization vector quantization (FSAPSOVQ) learning schemes, combined advantages of the fuzzy inference method (FIM), the simple VQ concept and the efficient self-adaptive particle swarm optimization (SAPSO), are considered at the same time to automatically create near optimum codebook to achieve the application of image compression. The proposed scheme uses the self-adaptive strategy that offers better efficiency in terms of convergence speed and also preventing the global best particle from getting stuck in local optima as in the particle swarm optimization. SAPSO algorithm is having two states (exploration and exploitation states) for each particle and a metric to measure a particle’s activity is defined which is used to choose which state it would reside. In order to balance a particle’s exploration and exploitation capability for different evolving phase, a self-adjusted inertia weight which varies dynamically with each particle’s evolution degree and the current swarm evolution degree is introduced into SAPSO learning algorithm. This FSAPSOVQ strategy is then compared with FPSOVQ and Linde- Buzo-Gray (LBG) algorithms to show its efficiency for different standard test images. Peak-signal-to-noise ratio (PSNR) is taking as a parameter to show the efficiency of above mentioned VQ.

Keywords: fuzzy inference analysis; self-Adaptive particle swarm optimization; vector quantization; image compression

1 Introduction

The objective of digital image compression techniques is the minimization of the number of bits required to represent an image, while maintaining acceptable image quality. The size of the image is determined by the resolution for each pixel for example a 512 pixel x 512 pixel having a resolution of 8 bits/pixel will have a size of 256 KB. The search for efficient image compression techniques ensues owing to the increasing pressure on transmission and storage media due to emergence of high definition multimedia. Vector quantization (VQ) is a very popular lossy data compression method that has seen numerous implementations in image compression systems owing to its simplicity. Recently, image compression is an important methodology to reduce the large amounts of handling image data; such technology can improve the working speed in the transmission and storage of the popular multimedia, video, medical diagnosis and interpretation system. The goal is to obtain an optimum codebook of finite length which can faithfully represent original image. Once such an optimum codebook is obtained the training process is finished. The encoding process of VQ with a selected code vector is to determine the mapping of input image set into finite collection of codebook. When all image training patterns are marked with their relative index to the nearest code vector, the encoding phase is finished. Such codebook is much smaller than the original image data set. Therefore, the purpose of image compression is achieved. In decoding process, the associated sub-image is exactly retrieved by the same codebook which has been used in the encoding phase. When each sub-image is completely reconstructed, the decoding phase is completed. One of the desirable objectives of the VQ compression technology is to increase the compression rate and achieve a higher fidelity. The higher of the compression rate, decreases the more required memory and the transmission channel bandwidth. A good image compression skill is

* Corresponding author. E-mail address: itsmemanishkumar@rediffmail.com, itsmemanishkumar@gmail.com

Copyright © World Academic Press, World Academic Union
IJNS.2010.06.15/352
not only achieving the highest compression rate but also assuring the good quality of the compressed image. The most well-known LBG vector quantization algorithm is an iterative learning based VQ method to approach optimal codebooks. LBG algorithm is a Generalized Lloyd type approach for codebook construction. It initializes M code vectors which are first created randomly, which are then processed iteratively using Euclidean norm type metric to yield a codebook that minimizes the average distortion between the training patterns and code vectors. Such an approach suffers from two major drawbacks; firstly the performance is much sensitive to choice of initial codebook as well as to the parameters. Secondly, since the Euclidean type metric is not convex, there is a possibility of being stuck in local minima in the codebook design. The hard decision approach imposed by LVQ fails upon the real images comprising of soft patterns. A soft decision scheme based on fuzzy sets is proposed with a Gaussian membership function to estimate the closeness of training vectors to the code vectors. The fuzzy particle swarm optimization (PSO) learning algorithm, combined the fuzzy inference analysis with PSO learning scheme, was proposed [10].

Attempts have been made to improve the PSO performance in recent years. Much work focused on parameters settings of the algorithm and on combining various techniques into the PSO. This paper explores for VQ an alternative called SAPSO algorithm in which there exist two states for each particle, the explorative state and the exploitative state. In the explorative state, it is attracted by the current global best position and its own personal best position. In the exploitative state, it is repelled away from its current personal best position and its personal worst position to search the other promising area. So, each particle is influenced not only by the current global best position and personal best position, but also its personal worst position that mainly depends on its current status. We also used a dynamic adaptation technique for SAPSO models where the inertia weight for each particle varies dynamically with its evolution degree and the current swarm evolution degree. PSO surfers from a much lower speed, such a lower convergence speed can be reflected by the lower accuracy. The proposed FSAPSO based learning algorithm in the design of optimum codebook of VQ to build up the image compression system in illustrated in Figure 1. The novel FSAPSOVQ learning algorithm, combined the fuzzy inference analysis with SAPSO learning scheme, is proposed to overcome the local minima of FPSOVQ learning algorithm which detects codebooks design to build image compressed system. Our experimental results demonstrate that FSAPSO outperforms in the comparison FPSO, in terms of solution accuracy, convergence speed and algorithm reliability.

2 Fuzzy self-adaptive particle swarm optimization vector quantization (FSAPSOVQ) codebook design

Codebook design is an important step in the design of vector quantization. VQ can be considered as a mapping procedure with the transmitted function of \( Q \) to convert \( n \)–dimensional Euclidean space \( R^n \) to a finite subset \( C \) of \( R^n \):

\[
Q : R^n \rightarrow C
\]  

(1)

where \( C = \{C_1, C_2, \ldots, C_M\} \) is the codebook with size \( M \) and each possible mapped code vector \( c_j = \{c_{j1}, c_{j2}, \ldots, c_{jn}\} \) in \( C \subset R^n \) is of \( n \)–dimension. Consider a \( n \)–dimensional problem where \( X = (x_1, x_2, \ldots, x_n) \) is an input training vector. In this FSAPSOVQ, fuzzy partition with the flexible membership function is to evaluate the performance of the selected code vector. Gaussian type fuzzy membership function is suggested to develop the
soft decision scheme and to process the fuzzy inference system. Then rules [9] are utilized to process the selection of the active code vector. Thus, the \( j \)th fuzzy rule can be stated as:

\[
R_j : \text{If } x_1 \text{is } HC_{(j,1)} \text{and} \ x_2 \text{is } HC_{(j,2)} \ldots \text{and} \ x_n \text{is } HC_{(j,n)} \text{Then } X \text{belong to } C_{(j,1,2,\ldots,n)} \text{with}
\]

\[
CF = CF_{(j,1,2,\ldots,n)}, \ j \in \{1, 2, \ldots, M\}
\]  

(2)

where \( n \) and \( M \) are the number of fuzzy sets and number of fuzzy rules respectively. \( HC_{(j,i)} \) is the \( i \)th fuzzy set for \( j \)th rule. \( C_{(j,1,2,\ldots,n)} \) determines the selected classification of code vectors and the \( X \) will be assigned to map to the proper \( C_{(j,1,2,\ldots,n)} \)th code vector with the grade of the \( CF_{(j,1,2,\ldots,n)} \) which denotes the certainty of the corresponding fuzzy rule \( R_j \). The membership function of fuzzy set \( HC_{(j,i)} \) for the \( j \)th fuzzy rule is given as:

\[
HC_{(j,i)} (x_t, c_j, r) = \exp \left( - \left( \frac{(x_t - c_{ji})^2}{r^2} \right) \right)
\]  

(3)

where

\[
r = \sqrt{\frac{\sum_{i=1}^{N_t} x_i - X}{N_t}}
\]  

(4)

and

\[
X = \frac{1}{N_t} \sum_{i=1}^{N_t} x_i
\]  

(5)

where \( N_t \) is the number of training patterns. The calculated membership value with equation (3), presents the evaluated grade of the selected input training pattern with respect to the selected \( j \)th code vector. Based on the same concept, the membership function indicates the soft evaluator to decide the similarity between input data \( x_t \) and \( j \)th code vector. The maximum grade of the membership function value is obtained while this data is the closest input pattern to the code vector. These contours of these membership functions are consider with feasible shapes and different center positions to approximate proper fuzzy partitions as soft evaluators of a fuzzy inference system to design the codebook of the VQ. When the \( t \)th training vector \( X_t = (x_{t1}, x_{t2}, \ldots, x_{tn}) \) is applied to the premise part of fuzzy inference procedure, the product operator is utilized to get the firing degree of the \( \{1, 2, \ldots, n\} \)th fuzzy sets with respect to the \( j \)th fuzzy rule, which is calculated as the following formula:

\[
\mu_j^t = HC_{(j,1)} (x_{t1}) \times HC_{(j,2)} (x_{t2}) \times \ldots \times HC_{(j,n)} (x_{tn})
\]  

(6)

In the case, the satisfying value of the certainly \( CF_{(j,1,2,\ldots,n)}^t \) is discussed as follows:

\[
CF_{(j,1,2,\ldots,n)}^t = \frac{(\mu_j^t - \alpha)}{\sum_{j=1}^{M} \mu_j^t}
\]  

(7)

where

\[
\alpha = \sum_{j=C_{(j,1,2,\ldots,n)}^t}^{N} \frac{\mu_j^t}{(N-1)}
\]  

(8)

While all input training patterns are applied to complete this fuzzy inference procedure, we obtained each certainty value \( CF_j^t \) for its associated \( j \)th code vector. Hence, the choice of the \( C_{(j,1,2,\ldots,n)}^t \)th active code vector is that it has the maximum \( CF_j^t \) value, the calculation is given by the following formula:

\[
CF_j^t = \max \left( CF_1^t, CF_2^t, \ldots, CF_M^t \right)
\]  

(9)

Based on formula (9), the \( x \)th code vector has the maximum value \( CF_x^t \), and then the input training pattern \( X_t \) will be assigned to the \( X^t \)th code vector. The evaluated scheme of the FSAPSOVQ input pattern compressed system in the fuzzy inference analysis that can be utilized to adaptively measure the variant input pattern. In general, this estimated input training pattern is the closest to the code vector; the maximum of the certainty value (CF) is to be achieved. In a word, the grade of the certainty value with the calculation of the fuzzy inference analysis indicates the similarity between selected input training patterns and code vectors. To efficiently approach the optimal codebook, suitable parameters of the fuzzy inference system with the guide of fitness function are selected by the SAPSO learning algorithm. The fitness function
of the SAPSO is defined to evaluate the performance of each individual solution. The objective of the fitness function is to correctly measure the distortion between the input data and the codebook. The definition of the fitness function is described as follows:

\[
\text{Fitness} = \frac{k_0}{\sum_{i=1}^{N_t} \sum_{j=1}^{M} s_{ij} \cdot \| x_i - c_j \|^2}
\]

where \(N_t\) is the number of training patterns, \(k_0\) assigned by the designer is a positive constant and the definition of \(s_{ij}\) is given by

\[
s_{ij} = \begin{cases} 
1 & \text{if } CF_{ij} = \max \{CF_1, CF_2, ..., CF_N\} \\
0 & \text{otherwise}
\end{cases}
\]

Here, \(x_i\) and \(c_j\) are the values of the \(i^{th}\) original input pattern and the selected \(j^{th}\) code vector, respectively. It is clear from (10) and (11) that the smaller actual error between the original input patterns and the code vectors will approach the smaller distortion. Therefore, the design in achieving appropriate codebook in the searching space can be formulated as the following searching problem:

\[
\min \left\{ \sum_{i=1}^{N_t} \sum_{j=1}^{N} s_{ij} \cdot \| x_i - c_j \|^2 \right\} = \max \left\{ \frac{k_0}{\sum_{i=1}^{N_t} \sum_{j=1}^{N} s_{ij} \cdot \| x_i - c_j \|^2} \right\}
\]

A large fitness value indicates that the behavior of selected fuzzy inference system will be characterized well as a codebook to approach minimal distortion. The best global particle which contains the highest fitness value in the whole swarm is called the global best (gbest) in SAPSO learning algorithm. The other important considered information, called personal best (pbest), is determined with evaluating each particle’s best solution so far.

---

**Figure 3:** Original training image (a)Lena (b)Pappers

### 2.1 Self-Adaptive particle swarm optimization

Particle swarm optimization (PSO) is one of the swarm intelligence (SI) algorithms that were first introduced by Kennedy and Eberhart, inspired by swarm behaviors such as birds flocking and fishes schooling [11]. In PSO, an underlying relation exist among the inertia weight, fitness, swarm size and dimension size of solution space, which could be used for accelerating the convergence of the PSO. Each particle’s inertia weight should be self-adjusted according to its own evolution degree and the current swarm evolution degree, which leads to the self-adaptive PSO. Self-Adaptive PSO (SAPSO) uses two main states for each particle, the exploitative state and explorative state. In the exploitative state, particle is attracted by its personal best position and current global best position like the original PSO algorithm. In the explorative state, particle is repelled away from its personal best position and personal worst position to search the un-reached regions. At each updating cycle in PSO algorithm [11], the velocity \(v_{ji}\) of the particle is regulated according to the relative personal and global information of gbest\(_{ji}\) and pbest\(_{ji}\). The PSO-based learning formula is defined as follows:

\[
v_{ji}(t+1) = \tau \cdot v_{ji}(t) + \beta_1 \cdot \text{rand()} \cdot (\text{pbest}_{ji} - Y_{ji}(t)) + \beta_2 \cdot \text{rand()} \cdot (\text{gbest}_{ji} - Y_{ji}(t))
\]

**IJNS email for contribution:** editor@nonlinearscience.org.uk
where $Y = \{c_1, c_2, .., c_N\}$ is the codebook with size $N$, $i$ is the index of codebook’s dimensions, $j$ is the index of the codebook’s number (i.e. particle’s index), $\tau$ is the inertia factor, $t$ employs current state and $(t + 1)$ describes the next time step, $\beta_1$ and $\beta_2$ are the parameters to control the influence of the cognitive and social learning rates, respectively. While the current velocity is decided, the $Y_{ji}$ in the next time step in PSO is moved and updated by

$$
Y_{ji}(t + 1) = Y_{ji}(t) + v_{ji}(t + 1)
$$

From equations (13) and (14), the learning path and the moving moment for each particle in PSO algorithm will be adjusted by the influence of both the global best solution (gbest) and the personal best solution (pbest). There exists an underlying relation among the inertia weight, fitness and dimension of solution space, which could be used for accelerating the convergence of the PSO. Specifically, define a self-adaptive inertia weight function for a particle in terms of its fitness value, the swarm size and bustle function as in (17). Define a bustle function for a particle in terms of dimension and particle fitness function as in (15). In order to decide which state to stay for a particle at time $t$, a particle bustle metric is defined which can reflect its current exploration capability. Here, in self-adaptive PSO a switching strategy is used between exploration and exploitative state. Bustle for every particle is computed firstly before updating a particle. If bustle is smaller than the threshold which adjusts self-adaptively with time, use the equation (13) to update its velocity and it is in the explorative state. Otherwise, it stays in the exploitative state and its velocity is updated by original PSO. This probabilistically divergent behavior can have a positive influence on the diversity of the solution, thereby improving its exploration capabilities. This is very much like the nature works. When the birds acquire all the food of a place, they fly away to search the places they have never reached. The particle bustle metric is defined for a particle $P_i$ at time step $t$, its activity is computed as equation (15):

$$
B(P_i(t)) = \sum_{d=0}^{D-1} \left| p_{\text{best}}(t)[d] - p_{\text{gbest}}(t)[d] \right| / D \cdot L[d]
$$

where $D$ is the dimensionality of the problem and $L[d]$ is the length of the $d$th dimension. Bustle value of a particle varies in the range of $[0, 1]$. Furthermore, the threshold of each particle varies with time dynamically. At time step $t$, it is computed as follows:

$$
a(t) = \left\{ \begin{array}{ll}
{t_{\max} - t} & \text{if } t < t_{\max} \\
n \cdot (a_i - a_f) + a_f & \text{if } t \geq t_{\max}
\end{array} \right.
$$

where $a_i$ is the initial threshold value at the start of a given run, and $a_f$ is the final threshold value at the end of a given run when $t = t_{\max}$. Maximum evolution number in a given run, current evolution number and nonlinear modulation index are denoted as $t_{\max}$, $t$, and $n$ respectively. Obviously, the system should start with a high initial threshold value for coarse global exploration and the threshold should linearly decrease to facilitate finer local explorations in later iterations. This should help the system to approach the optimum of the fitness function quickly. It can be seen that when $n = 1$ the threshold varies linearly adaptive with time, otherwise it is varied dynamically as a nonlinear function of the present iteration number at each time step. The most important parameter that drives the behavior of PSO towards the convergence is inertia weight, which was introduced by Shi and Eberhart to eliminate the need for velocity clamping, but to still restrict divergent behavior [12, 13]. It controls the momentum of the particle by weighing the contribution of the previous velocity. While static inertia values have been used successfully, adaptive inertia values have also shown to lead to convergent behavior [14-16]. Each particle start with a high inertia weight (causes higher changes in velocity per time step) for coarse global exploration. During the search process, each particle’s inertia weight should be self-adjusted according to its own evolution degree and the current swarm evolution degree. Based on this, the inertia weight for particle $p_i$ at time $t$ is given by formulas (17):

$$
w_i(t) = 0.15 + \frac{1}{1 + e^{a_v - f(p_{\text{best}}(t))}}
$$

where $a_v(t) = \sum_{i=0}^{N-1} f(p_{\text{best}}(t))/N$ which can reflect the swarm average evolution degree at time $t$. So the velocity updating rule for particle $p_i$ at time step $t$ becomes formula (21) for explorative state and formula (22) for exploitative state.

$$
\begin{align*}
v_{ji}(t + 1) &= \tau_{ji}(t) \cdot v_{ji}(t) - \beta_1 \cdot \text{rand}() \cdot (p_{\text{best}}_{ji} - Y_{ji}(t)) - \beta_2 \cdot \text{rand}() \cdot (p_{\text{worst}}_{ji} - Y_{ji}(t)) \\
v_{ji}(t + 1) &= \tau_{ji}(t) \cdot v_{ji}(t) + \beta_1 \cdot \text{rand}() \cdot (p_{\text{best}}_{ji} - Y_{ji}(t)) + \beta_2 \cdot \text{rand}() \cdot (g_{\text{best}}_{ji} - Y_{ji}(t))
\end{align*}
$$

Obviously, this can make the particle with high evolution degree emphasize particularly on fine turning exploitation ability to improve solution quality, and the particle with low evolution degree possess larger momentum for exploration.

IJNS homepage: http://www.nonlinearscience.org.uk/
This helps to approach the optimum of the fitness function quickly. From equation (13), it can be easily obtained that for \(\tau > 1\), velocities increase over time causing divergent behavior. Particles fail to change direction in order to move back towards promising areas. For \(\tau < 0\), particles decelerate until their velocities reach zero. To ensure convergence, the value for \(\tau\) should be chosen so that it satisfied the following relation.

\[
\begin{align*}
\tau > & \frac{1}{2} (\beta_1 + \beta_2) \\
0 < & \tau < 1 
\end{align*}
\]

This relation can also be reversed to calculate the values of \(\beta_1\) and \(\beta_2\) once a suitable \(\tau > 1\) has been decided on. In our proposed SAPSO algorithm, despite the inertia weight for each particle varies self-adaptively, its value will vary in the range of \((0, 1)\) in later generations according to equation (17). When all particles achieve this, the swarm does not change any more which is global best position will not change either. Thus, it can be stated that SAPSO algorithm possesses the property of convergence.

3 Proposed algorithm

In FSAPSOVQ, each of the particles in the swarm is classified in one of the two states exploration and exploitation states. The value of social and cognitive learning rates is calculated for each particle using this state information contained in the evolutionary factor. The inertia weight is also adjusted according to the equation (20). The procedure of learning from the input pattern is summarized in the following steps:

1. Randomly initialize parameters of the fuzzy inference system \((Y_{ji})\) and its associated velocity \(v_{ji}\) for all particles. Here, \(Y_{ji}\) are adjustable parameters set with respect to the jth particle.

2. Apply the initial solution \(Y_{ji}\) to generate fuzzy inference system. Apply initial original image data set to process the fuzzy IF-Then rules to generate the grade of the CF and assign the image data to belong to the code vector which has maximal CF value.

3. Calculate the fitness value of each particle with Eq. (10) and select the personal best (pbest) of each particle by

\[
p_{best_j}(t+1) = \begin{cases} 
Y_{ji} (t+1) & \text{if } \text{Fitness}(Y_{ji} (t+1)) \geq \text{Fitness}(Y_{ji} (t)) \\
Y_{ji} (t) & \text{if } \text{Fitness}(Y_{ji} (t+1)) < \text{Fitness}(Y_{ji} (t)) 
\end{cases}
\]

4. Select global best position (gbest) of each particle using (22)

\[
gbest = \text{Arg max } \{\text{Fitness}(p_{best_j}(t+1))\}
\]
Figure 4: Testing results with Lena image (a) a 256*256 Lena Image (b) Histogram from original Lena image (c) LBGVQ reconstructed image of Lena (M=256, PSNR=32.3532) (d) Histogram from LBGVQ reconstructed image, (e) FPSOVQ reconstructed image of Lena (M=256, PSNR=36.3078), (f) Histogram from FPSOVQ reconstructed image, (g) FSAPSOVQ reconstructed image of Lena (M=256, PSNR=43.0436), (h) Histogram from FSAPSOVQ reconstructed image.

5. Calculate the distance of each particle from all the particles using

\[ d_i = \frac{1}{M-1} \sum_{j=1, j \neq i}^{M} \sqrt{\sum_{k=1}^{n} (x_k^i - x_k^j)^2} \]  
(23)

6. Calculate the bustle value from equation (15) and threshold from (16). Decide the state of particles according to above said strategy.

7. Calculate the inertia factor using (17).

8. Update the velocities and position of particles using Eq. (18) or (19) depending up on the state of the particle.

9. Repeat steps 2-8 until a stop criterion is met or a predefined number of iterations is completed.

10. The optimal codebook is constituted by proposed fuzzy inference analysis with global parameters set (gbest).

11. Transmit the optimal codebook to the decoding procedure and rebuild the compressed image.

4 Results

Lena and peppers are selected as the training images which are shown in figure 3. They are gray scale 256x256 pixel images. In FAPSOVQ, parameters definitions are swarm size = 20, \( C_1 = C_2 = 1.5 \) and \( k_0 = 50 \). The peak signal to noise ratio (PSNR) is proposed to evalute the quality of the reconstructed image. The PSNR is defined as follows:

\[ PSNR = 10 \log_{10} \frac{255^2}{\frac{1}{N \times N} \sum_{i=0}^{N-1} \sum_{j=0}^{N-1} (f(i,j) - \hat{f}(i,j))^2} \text{dB} \]  
(24)

where \( N \times N \) is the size of the original image and \( f(i,j) \) and \( \hat{f}(i,j) \) are the gray level pixel value of the original image and the reconstructed one, respectively.

At first experiment, two original images, lena and peppers are considered to test the adapt capacities between LBG, FPSO, FSAPSO learning methods of VQ machine with different sizes of codebooks, such as \( M = 4, 8, 16, 32, 64, 128, 256 \). After running FSAPSO, LBG, and FPSO learning algorithm three times, Table 1 of the PSNR against various sizes of codebooks for Lena and Pepers are shown. In this simulation, it is clear that the three generated results with the same LBG-based learning scheme are highly affected by the variation of initial conditions. The other obvious circumstance of this representation in our simulation is that the difference in performance comparison between LBG, FPSO and FSAPSO will gradually become large when we increase the size of the codebook. The proposed SAPSO learning scheme can avoid the local minima of the PSO and keep the higher PSNR value even if it meets the different sizes of the codebook. The three generated results by means of running FSAPSO learning methods almost have similar good curves, it describes the
Figure 5: Testing results with Peppers image (a) a 256*256 Peppers Image (b) histogram from original Peppers image (c) LBGVQ reconstructed image of Peppers (M=256, PSNR=31.7226) (d) histogram from LBGVQ reconstructed image, (e) FPSOVQ reconstructed image of Peppers (M=256, PSNR=35.3088), (f) histogram from FPSOVQ reconstructed image, (g) FSAPSOVQ reconstructed image of Peppers (M=256, PSNR=40.9419), (h) histogram from FSAPSOVQ reconstructed image.

ability of FSAPSOVQ learning scheme, that has powerful and adaptive ability, to deal with the initial problem in the design of complex codebooks where LBG fails and also having ability to jump from the local minima where FPSOVQ learning scheme stuck up which increase the performance of FSAPSOVQ learning scheme drastically.

5 Conclusion

In this article, the evolution FSAPSO learning scheme is proposed to find optimal parameters of fuzzy inference system, and then are generated approipriate codebooks with the decision of the soft fuzzy inference analysis to achieve the application of image compression. Based on the self adaptive particle swarm optimization learning scheme and flexible membership function of fuzzy inference system, the simulation demonstarated advancement of the FSAPSOVQ-based image compression system. It is obvious that proposed FSAPSO learning method can solve the local minima problem of FPSO and LBG learning schemes. Two standard test images Lena and Peppers are regarded as the original testing patterns to verify that the FSAPSO learning algorithm has better global and robust performances than FPSO and LBG learning scheme. In real word image application, the processing information is usually vague, variant and so on. In our experiments, efficient FSAPSOVQ learning scheme is self adaptive enough to suffer such vague and variant enviroment and design good codebooks to reconstruct the compressed image with high fidelity.

References


IINS email for contribution: editor@nonlinearscience.org.uk


IJNS homepage: http://www.nonlinearscience.org.uk/