Non-Parametric Vector Quantization Algorithm

Haemwaan Sivaraks and **Athasit Surarerks**, Non-members

ABSTRACT

Recently researches in vector quantization domain concentrate on improving the performance of parametric algorithms (i.e., need to specify either the codebook size or the expected distortion.) The specification of the both parameters gives the users some difficulties. We, in the research, propose a non-parametric vector quantization (NVQ) algorithm. The concept is to manage Voronoi regions with respect to the distortion decreasing rate. Experimental results point that the average distortions are statistically decreased comparing with LBG and adaptive incremental LBG algorithms at the 99% of confidence level. Both speech and image data are also included in our consideration.

Keywords: Vector Quantization, codebook, codeword, Non-parametric Quantization

1. INTRODUCTION

Vector Quantization (VQ) [1, 2] is considered as a lossy data compression method. In detail, all vectors are grouped by some geometric properties, and the centric of each group is promoted to be a codeword. The obtained codebook (the collection of all codewords) is said to be the quantized (or compressed) data which usually use for representing the data. The performance of the codebook can be measured by its distortion.

Quantization technique is widespread in many applications [3, 4], such as image compression, speech compression, image retrieval and pattern recognition. VQ can reduce the data size while producing only little distortion from real data. Therefore, the results of VQ, or codebooks, can be used as a representation of real data. Advantages of using codebooks are to reduce the storage data and computation time.

Many algorithms have been proposed bases on the Kmeans [5] idea, such as LBG algorithm [6], ELBG algorithm [7, 8] and adaptive incremental LBG [9]. The results of these algorithms depend on the input parameters specified by user, which are codebook sizes or distortion.

Manuscript received on December 14, 2006 ; revised on March 19, 2007.

The authors are with the Department of Computer Engineering, Chulalongkorn University Patumwan, Bangkok 10330, Thailand; E-mail: g48hsv@cp.eng.chula.ac.th, athasit@cp.eng.chula.ac.th

The codebook sizes or distortion have direct effects on the resulting codebooks. If the codebook size is too small or the distortion is too large, the resulting codebook will poorly represent real data. However, if the codebook size is too large or the distortion is too small, the resulting codebook size will be almost as large as the size of the real data.

Based on the mentioned problems, we propose a new algorithm called non-parametric vector quantization (NVQ). This proposed algorithm does not require user to define any input parameters by improving of LBG algorithm with automatic identifying codebook size. Our concept is to create both of codebook and its size in order to obtain the minimum distortion.

The paper is organized as follows. Section 2 describes some basic definitions including LBG algorithm, adaptive incremental LBG algorithm. An improvement of LBG algorithm with automatic identifying codebook size is also introduces in this section. Section 3 proposes a novel vector quantization algorithm named Non-parametric vector quantization (NVQ). Some experimental results are analyzed in Section 4. Conclusions are presented in Section 5.

2. VECTOR QUANTIZATION

This section presents basic concepts of vector quantization and introduces the traditional LBG algorithm, the adaptive incremental LBG algorithm, and the improvement of LBG algorithm with automatic identifying codebook size.

2.1 Definition

Vector quantization Q is the technique for representing a k-dimensional vector set$X = \{x_1, x_1, ..., x_m\}$ by a vector set $C = \{c_1, c_1, ..., c_n\}$,where k $\geq$ 2 and $m >> n$. So, a vector quantization can be represented as a function:

$$Q : X \longrightarrow C$$

The vectors of X are called the input data or the input vectors. C is said to be a codebook and its elements are called codewords. Associated with n codewords, there is a partition $R_1, R_1, ..., R_m$ for X, where

$$R_j = \{x \in X : Q(x) = c_j\} \text{ , j = 1,...,n}$$

All vector in X which have the same image of Q is said to be contained in the same region. From this definition, the regions are non-overlapping and their

union is X. A Group of data in the same region is represented by one codeword.

Vector quantization uses a distance function that measures the distance between one vector and one codeword. A commonly used distance function is the Euclidean distance, shown in equation (1)

$$d(x,c) = \sqrt{\sum_{i=1}^{k}(x_i - c_i)} \quad (1)$$

Distance is used to partition the region using the Nearest neighbor condition. The distance function is also used to calculate the distortion.

There are many criteria used to evaluate codebooks. Criteria selection depends on applications. General applications place importance on distortion. The lower the distortion, the better it is. The measure of distortion is the mean quantization error (MQE), shown in equation (2) and (3)

$$MQE \equiv D(\{C,R\}) = \frac{1}{np}\sum_{i=1}^{nc} E_i \quad (2)$$

$$E_i = \sum_{j:x_j \in R_i} d(x_j, c_i) \quad (3)$$

where E_i is the local quantization error (LQE) of codeword c_i.

The centroid measurement is used in the adjustment of codeword position. The codeword values at the centroid of region produces minimize distortion. The centroid value is calculated by equation (4)

$$c_{j,k} = \frac{1}{N_j}\sum_{i=1}^{N_j} x_{i,k}, \quad x_i \in R_j \quad (4)$$

where $N_j$is the number of vectors belonging toR_j.

The criteria used in vector quantization process is usually called the nearest neighbor condition (NNC). It is used to partition data into regions.

The partitionR_j,j=1,...,m must satisfy

$$R_j \supset \{x \in X : \forall i \;\; d(x,c_j) < d(x,c_i), j \neq i\}$$

where R_j is the region of codeword c_j, j=1,...,n.

2.2 Linde-Buzo-Gray algorithm

This algorithm proposed by Yoseph Linde, Andres Buzo and Robert M. Gray [6] in 1980. Their algorithm is known as the generalized Lloyd algorithm (GLA) or LBG algorithm. Many proceeding algorithms are developed based on this algorithm. This algorithm requires user to predefine the codebook size. The description of this algorithm is as follows.

Algorithm 2.2 LBG algorithm
Input: m input vectors, n codebook size
Output:codebook
1) Random n initial codebook.
2) Partition the input vector for each codeword, just like defining regions. The partition is calculated according to the nearest neighbor condition.
3) Check termination conditions. Compute the distortion and consider the change rate of distortion from previous iteration. If $(D_{prev} - D_{curr}/D_{curr} \leq \in$,terminate the algorithm. Otherwise, continue.
4) The new codebook is calculated according to the centroid measurement. Repeat step 2.

In every iteration, the codebook size is constant to what user predefine. The codebook size is inflexible. If the codebook size is too large some codeword probably represent empty regions as illustrated by Fig.1. This algorithm cannot remove these misleading codewords.

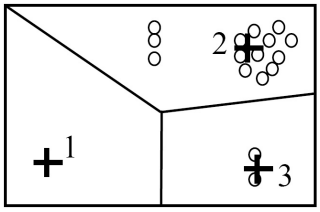


Fig.1: *Misleading Codewords.*

Furthermore, this algorithm adjusts the codebook by calculating from the centroid of each region. Thus, the resulting codebook depends on the initial codebook.

2.3 Adaptive incremental LBG

An adaptive incremental LBG [9] was proposed by Furao Shen and Osamu Hasegawa in 2006. This algorithm was developed based on the LBG algorithm in order to solve a problem of inflexible codebook size. This algorithm has two alternatives which user has to define: codebook size or distortion. Both alternatives follow similar procedures but use different criteria. The description of this algorithm is as follows.

Algorithm 2.3 Adaptive incremental LBG algorithm
input : m input vectors, n codebook size or ξ distortion
output : codebook
1) Initialize the codebook containing one codeword which is selected randomly from the original input vector set.
2) Execute the LBG algorithm to optimize the codebook.
3) If the current codebook size is smaller than n (or if the current distortion is greater than ξ), insert a new codeword to the codebook until the current codebook size is equal to n (or the current distortion is less than or equal to ξ).
4) If the distortion in the current iteration is greater than the distortion from previous iteration distortion (or the codebook size from current iteration is greater

than the codebook size from previous iteration), return the previous codebook as a result. Otherwise, remove the codewords whose LQE are 0 or the lowest. Back to step 3.

This algorithm still depends on user to determine input parameters, codebook size or distortion. A new codeword is inserted by selecting random value form the original data. In the worst case, a poorly represented codebook is possible if some codewords represent only one vector. Therefore, the algorithm must repeat the process of removing codewords again which adds unnecessary step to the algorithm.

2.4 An improvement of LBG algorithm with automatic identifying codebook size

This algorithm [11] solves the problem of LBG algorithm. It does not depend on user to predefine any input parameters, and produces a lower distortion of the resulting codebook. This algorithm is developed based on the LBG algorithm and uses the splitting method from the adaptive incremental LBG algorithm.

This algorithm uses adjacent codewords with the second shortest from the data. The adjacent codeword is denoted by *adjcw*. During the adjusting process of the regions, this algorithm calculates distortion in each subregions. This distortion is called inner distortion which is calculated by equation (5)

$$D_j = \frac{1}{N_j} \sum_{i=1}^{N_j} d(x_i, c_j, \ \forall x_i \in R_j \qquad (5)$$

where D_j is the distortion in partition R_j

This algorithm increases the ability of vector quantization. User need not determine input parameters, both codebook size and distortion. User can just input the data and the algorithm immediately process. This algorithm follows step in algorithm 3.1.

Algorithm 2.4 Improvement of LBG algorithm
input : m input vector
output : codebook
1) Initialize the codebook containing one codeword which is selected randomly from the original input vector set.
2) Partition the input vector for each codeword, just like defining regions. The partition is calculated according to the nearest neighbor condition.
3) Adjust the partitions using the following steps.

3.1) Find *maxcw*, which is the codeword with maximum inner distortion. Compute *maxcw* according to equation (5).

3.2) Find *adjcw* of maxcw.

3.3) Find *edgepoint*, which is the vector at the edge of each region. The *edgepoint* is represented by *maxcw* and is the closest to *adjcw*.

3.4) Transform the codeword of the *edgepoint*, *maxcw*, into *adjcw*.

3.5) Calculate a new codebook according to the centroid measurement.

3.6) If the inner distortion of **maxcw** is less than the distortion from the previous iteration, continue to step 4. Otherwise, use the value stored in a temporary variable, which calculates from the codebook in the previous iteration.
4) Compute the distortion and consider the change rate of distortion from the previous iteration.

If$(D_{prev} - D_{curr}/D_{curr} \leq 0.001$,continue with step 5. Otherwise, continue with step 3.
5) Partition and calculate a new codebook. The partition is calculated based on the nearest neighbor condition. The new codebook is calculated using the centroid measurement until the codebook does not change.

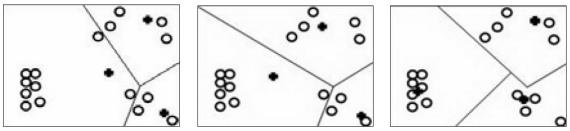


Fig.2: *Example of Partitions Adjustment.*

Fig. 2 shows steps in adjusting the codeword partitions by transforming the codeword of the vector at the edge of the region. This process increases one by one. This process increases computation cost. Moreover, an initial codebook is selected randomly from the original data. The random selection process results in inconsistent codebooks among each execution. This problem can be solved by the NVQ algorithm.

3. NON-PARAMETRIC VECTOR QUANTIZATION ALGORITHM

This proposed algorithm presents an non-parametric vector quantization algorithm, called NVQ algorithm. The outstanding characteristics of the NVQ are that user need not to determine any input parameters and that each time this algorithm is processed, it returns only one result. Therefore, NVQ is a deterministic algorithm. Many applications require the only one result to identify data; speaker identifying etc.

The NVQ algorithm uses iterative procedures to adjust the codebook. This algorithm initializes the codebook using the splitting method. The initial codebook consists of two codewords. After that, codewords are adjusted by shrinking and expanding the partition. The algorithm increases the codebook size using the splitting method.

Distortion evaluation uses both distortion of all data and distortion in each subregion, called inner distortion, which is calculated by equation (5).

The NVQ algorithm comprises of 3 main procedures as follows.

3.1 Splitting Codeword

The objective of codeword splitting is to increase the codebook size by one codeword. First, set the margins for data in the region that we want to increase the codeword by finding maximum and minimum data values for each dimension one by one. This process is shown in Fig. 3.

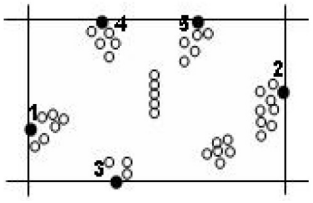


Fig.3: *Process of Setting Margins for Data in 2D*

Fig. 3 shows the process of setting margins for data in two dimensions. In practice, this technique can be used in multi-dimension. From this figure, the shaded circles show the maximum and minimum data values in each dimension, i.e., circle number 1 is the minimum value on the horizontal axis, circle number 2 is the maximum value on the horizontal axis, circle number 3 is the minimum value on the vertical axis and circle number 4 and 5 are the maximum value on the vertical axis.

There are two methods in splitting the codeword.

3.1.1 1-to-2 Splitting

The 1-to-2 splitting pattern. Composes of two steps.

a) Codebook Initialization. This step sets margins for all data.

b) Codeword Splitting when current codebook consists of 2 codewords. This step sets margin in the region with the maximum inner distortion.

After setting margins, the next step is to find possible positions for codewords as shown in Fig. 4.

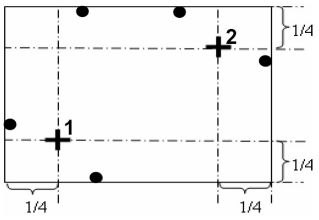


Fig.4: *Two Possible Positions for Codewords*

Fig. 4 shows the possible positions for codewords represented by the plus signs. They are obtained by calculating one-fourth of each margin on all dimensions. Values of two possible positions for codewords are the maximum data value minus one-fourth of the length of the margin and minimum data value plus one-fourth of the length of the margin in such dimension. After obtaining two possible positions for codewords, the next step is to find the nearest vector data to each possible position for codeword. The Two nearest vectors become new codewords.

3.1.2 2-to-3 Splitting

This method is used if the current codebook consists of more than 2 codewords. Process of setting the margins is different from the previous method. This method merges data in the maximum distortion region with data in the adjacent maximum distortion region before setting the margins.

The method of finding possible positions for codewords is show in Fig. 5.

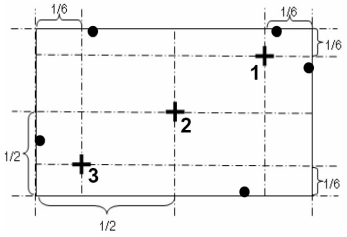


Fig.5: *Two Possible Positions for Codewords*

Fig. 5 shows three possible positions for codewords, represented by the plus signs. They are obtained by calculating one-sixth and one-half of each margin on all dimensions. Values of three possible positions for codewords are the maximum data value minus one-sixth of the length of the margin, the maximum data value minus ong-half of the length of the margin and the minimum data value plus one-sixth of the length of the margin in such dimension. After obtaining three possible positions for codewords, the next step is to find the nearest vector data to each possible position for codeword. The three nearest vectors become new codewords.

The ideas of both splitting methods are to spread out new codewords and to use the data nearest to possible positions for codewords in order to avoid misleading codewords.

3.2 Codebook adjustment

Codebook adjustment is the procedure to adjust the codeword positions in order to reduce distortion. There are two methods of adjustment as follows.

3.2.1 Partition shrinking and expanding

This method moves the adjacent codeword closer to the max codeword, which is the codeword with the maximum inner distortion. This process can be perceived as shrinking the maximum region, which is the region with maximum distortion, and expanding the adjacent regions in order to reduce distortion in the maximum region. The idea of this method is that the inner distortion affects all distortion. Consequently, we expect that by reducing the inner distortion, the overall distortion will be decreased too.

This procedure moves the adjacent codeword closer to maximum codeword by one-half of the length between those codewords in each dimension. This process is used because we want the adjacent codeword to cover the previous region. The last phase of procedure checks if the distortion of current codebook is not reduced, The previous codebook becomes the resulting codebook. However, if the distortion is reduced, continue using the current codebook. Examples of this process is shown in Fig. 6(a) - 6(c)

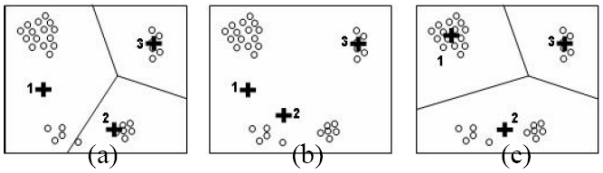


Fig.6: *(a) Before partition shrinking and expanding (b) Partition Shrinking and expanding (c) Partition and centroid calculation*

Fig. 6 shows examples of codeword adjustment by shrinking and expanding method. Fig. 6(a) shows data and codebook before partition shrinking and expanding process. Codeword number 1 is the maximum codeword. Codeword number 2 is the adjacent codeword. Fig. 6(b) shows the shifting of codeword number hs2 closer to codeword number 1. Fig. 6(c) shows region adjustment by using partition and centroid calculation, Description using this method is as follows.

3.2.2 Partition and centroid calculation

The codebook adjustment is the adjustment of each codeword to be closer to the centroid of each region, where the distortion is the lowest. First the region is partitioned based on the nearest neighbor condition. Second, the centroid partition is calculated for each region.

3.3 Criteria in NVQ

There are three main criteria in the algorithm.

3.3.1 Criteria for max codeword changing

Partition shrinking and expanding is processed until the inner distortion, which is the distortion in the region, is lower than the distortion that is calculated from all data. Then, this algorithm changes the codeword from max codeword to other codewords which have never been used as max codewords nor adjacent codewords in previous iterations using the same codebook size.

3.3.2 Criteria for codeword splitting

Codebook size incremental is used if the current codebook size is too small to terminate the algorithm. If the max codeword cannot be found, the algorithm splits the codeword to reduce the distortion.

3.3.3 Criteria for algorithm termination

Algorithm termination considers the change rate of distortion from the previous iteration. According to relative error[12] of distortion, if the changer rate is less than 0.001, the algorithm will be terminated. The change rate is computed as follow

$$\epsilon = \frac{|D_{prev} - D_{curr}|}{D_{curr}}$$

Description of the proposed algorithm is as follows.

NVQ Algorithm :
Input : a set of vector
Output : codebook C

1. Initialize codebook C based on the 1-to-2 splitting method.
2. Partition and calculate centroid using the current codebook. Partition the region according to the Nearest neighbor condition. Calculate centroid using centroid measurement.
3. Find the codeword with the maximum distortion.
4. Check codeword splitting condition.
5. Find the codeword adjacent to the maximum codeword.
6. Shrink and expand the partition.
7. Check criteria for max codeword changing. If the codeword meets the criteria, change max codeword and continue with step 8. Otherwise, vary the adjacent codeword during the process of shrinking and expanding the partition.
8. Repeat step 3-7 until the algorithm meets the termination criteria and continue with step 9.
9. Partition and calculate centroid using the current codebook until the codebook does not change.

The NVQ algorithm finds the codebook without user inputs on codebook size or distortion. This process is possible because the algorithm increases codebook size one by one. For each codebook size, the codeword position is adjusted to find the suitable codebook by shrinking and expanding the partition.

4. EXPERIMENT

This session presents the distortion obtained from the NVQ algorithm and compares it with the distortion from other algorithms, LBG algorithm and adaptive incremental LBG. The comparison is done using the codebook size from the NVQ algorithm. Most of vector data are generated by 2-dimension images and multidimension voice[19, 20]. The description about sets of vector data which are generated by image and random vectors is as follows.

1. Plane image 7,036 vectors, illustrated by Fig. 7
2. Lena image, a popular test image, 24,289 vectors, illustrated by Fig. 8
3. Baboon monkey image 16,906 vectors, illustrated by Fig. 9

4. Cluster image 390 vectors, illustrated by Fig. 10
5. Text image 3,376 vectors, illustrated by Fig. 11
6. Curve image 1,591 vectors, illustrated by Fig. 12
7. Unstructured vector data 5,627 vectors, illustrated by Fig. 13
8. Random vector 300 vectors, illustrated by Fig. 14
9. Oval image 416 vectors, illustrated by Fig. 15
10. Heart image 395 vectors, illustrated by Fig. 16

Description about sets of vector generated by voice is shown in Table 1

Table 1: *Description of voice data*

Voice set	Dimensions	Vector size
1	5D	988
2	10D	988
3	15D	988
4	20D	988
5	5D	1,197
6	10D	1,197
7	15D	1,197
8	20D	1,197

We tested the NVQ algorithm first because the codebook size from NVQ will be used by other algorithms. This experiment compares average distortion of LBG and adaptive incremental LBG with the distortion of NVQ. The distortion is used for performance evaluation because general applications in vector quantization use this value to determine the suitability of the codebook. We shown the distortion of codebooks obtained from our algorithm and the average distortion from other algorithms in Table 2.

Table 2 shows the comparison of distortion between NVQ algorithm and other algorithms. The distortion of NVQ algorithm are less than other algorithms in all tests. Therefore, the proposed algorithm can produce the codebook with lower distortion. Moreover our algorithm can process vector quantization without user inputs on codebook value and distortion. User can only input the data and our algorithm automatically processes the data.

In order to test distortion from different algorithms, we test the differences using the paired T-test. The NVQ algorithm, when compared with LBG and adaptive incremental LBG algorithms, shows statistically [21] better average distortion at the 99% confidence level. (tvalue = -3.639 and -4.235 respectively, p-value = 0.01)

5. CONCLUSION

Vector quantization is one method that can reduce data size. User can use small codebooks instead of using large data. But, currently, several VQ algorithms depend on user to predefine codebook sizes or distortion as input parameter. Thus, the resulting codebook directly depends on user inputs. Consequently, we developed an Nonparametric vector quantization

Fig 7: *Testing set 1*

Fig 8: *Testing set 2*

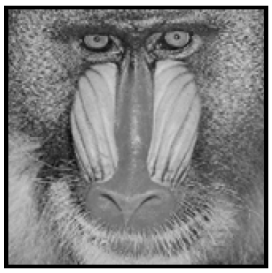


Fig 9: *Testing set 3*

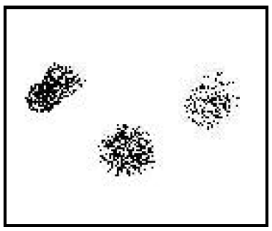


Fig 10: *Testing set 4*

Fig 11: *Testing set 5*

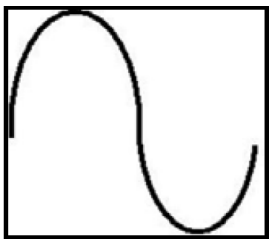


Fig 12: *Testing set 6*

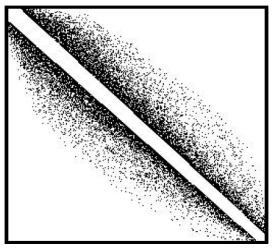


Fig 13: *Testing set 7*

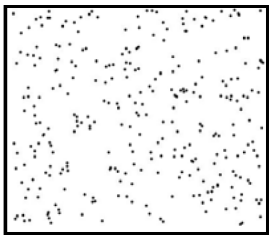


Fig 14: *Testing set 8*

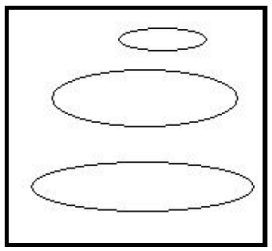


Fig 15: *Testing set 9*

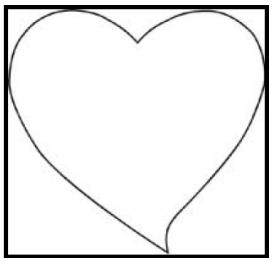


Fig 16: *Testing set 10*

Table 2: *Distortion comparison for three algorithms*

Data set	NVQ algorithm	LBG algorithm	Adaptive incremental LBG	Fixed codebook size
Image 1	21.20570	21.91676	22.96681	8
Image 2	24.41400	25.69475	26.20939	11
Image 3	10.33940	10.50260	10.74867	6
Image 4	6.07610	6.977175	6.11177	6
Image 5	14.49080	16.07851	14.92430	9
Image 6	23.23310	23.57405	24.88860	6
Image 7	28.19770	28.30011	28.27454	5
Image 8	12.20870	12.55355	13.27508	9
Image 9	15.76310	15.89804	16.11379	8
Image 10	15.85310	16.57947	16.80970	9
Voice1	7.9384	8.02304	8.23042	11
Voice 2	8.4943	8.64012	8.92075	9
Voice 3	14.3928	14.5448	14.6247	8
Voice 4	15.8744	15.9853	16.2588	5
Voice 5	19.4307	19.5735	19.5969	5
Voice 6	18.839	18.8482	19.0383	6
Voice 7	21.001	21.038	21.0832	3
Voice 8	20.04	20.1652	20.3535	4

to solve this problem. We proposed the NVQ algorithm which is a deterministic algorithm. Techniques of this algorithm are shrinking and expanding the partition using the same codebook size and increasing the codebook size when the current codebook cannot reduce the distortion. This proposed algorithm can process vector quantization automatically after the data are input. User need not concern about input parameter values. Moreover, from the experiment, NVQ algorithm gives lower average distortion than LBG algorithm and adaptive incremental LBG.

References

[1] R. M. Gray, Vector Quantization, *IEEE ASSP Magazine*, Vol.1, pp.4-29, 1984.

[2] A.Nongnuch and A.Surarerks, A Novel Approach of Density Estimation for Vector Quantization, *WSEAS Transactions on computers*, Vol.9, pp.1179-1186, 2005.

[3] A,Nongnuch, *A Novel Approach of Density Estimation for Vector Quantization*, M. Thesis, Chulalongkorn University, Bangkok, Thailang, 2005.

[4] G. Patane, *Unsupervised Learning on Traditional and Distributed Systems*, D. Eng. Thesis, Palermo, Italy, 2001.

[5] J.Han and M. Kamber, *Data Mining Concepts and Techniques*, Morgan Kaufmann Publishers, United States of America, 2001.

[6] Y. Linde, A. Buzo and R.M. Gray, An algorithm for vector quantization design, *IEEE Transaction on communications*, Vol.COM-28, No.1, pp.84-95, 1980.

[7] G. Patane and M. Russo, ELBG Implementation, *International Journal of Knowledge based Intelligent Engineering Systems*, Vol.4, pp.94-109, 2000.

[8] G. Patane and M. Russo, The enhanced LBG algorithm, *Neural Networks*, Vol.14, pp.1219-1237, 2001.

[9] F. Shen and O. Hasegawa, An adaptive incremental LBG for vector quantization, *Neural Networks*, Vol.19 Issue 5, pp.694-704, 2006.

[10] G. Patane and M. Russo, Fully Automatic Clustering System, *IEEE Transactions on Neural Networks*, Vol.13, pp.1285-1298, 2002.

[11] H.Sivaraks, A.Surarerks, An improvement of LBG algorithm with automatic identifying codebook size, *Proceeding of 10th National Computer Science and Engineering Conference (NCSEC2006)*, pp.103-110, 2006.

[12] C. C. Steven, *Numerical Methods for Engineers*, McGraw-Hill, Singapore, 2006, ch. 3.

[13] O. N. Gerek and H. Cinar, Segmentation based coding of human face images for retrieval, *Signal Processing*, Vol.84, pp.1041-1047, 2004.

[14] A. Orlitsky, Scalar vs vector quantization worst-case analysis, *IEEE Transactions on Information Theory*, Vol.48, pp.1393-1409, 2001.

[15] K. Somasundaram and S. Domnic, Modified Vector Quantization Method for Image Compression, *Transactions on Engineering Computing and Technology*, Vol.13, pp.222-227, 2006.

[16] W. A. Equitz, A new vector quantization clustering algorithm, *IEEE Transactions* on Acoustics Speech and Signal Processing, Vol.37 pp.1568-1575, 1989.

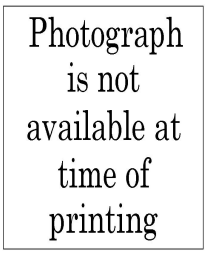
[17] O. Virmajoki, P. Franti and T. Kaukoranta, Iterative shrinking method for generating clustering, *IEEE International Conference on Image Processing*, pp.685-688, 2002.

[18] R. Xu, Survey of Clustering Algorithms, *IEEE Transactions on Neural Networks*, Vol.16, No.3, pp.645-678, 2005.

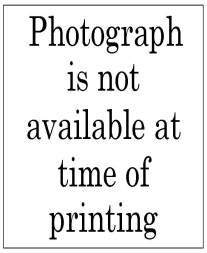
[19] W. Phil, *Hidden Markov Model Toolkit* [Computer program], Cambridge University Engineering Department (HTK3), 2002, Available from: http://htk.eng.cam.ac.uk [2006, October 7]

[20] S. Kasuriya, V. Sornlertlamvanich, P. Cotsomrong, S. Kanokphara and N. Thatphithakkul, *Thai Speech Corpus for Thai Speech Recognition* [Computer data file], National Electronics and Computer Technology Center, 2003, Available from: http://vaja.nectec.or.th: 8083/lotus/ [2006, November 1]

[21] K. Vanichbuncha, *SPSS for Windows*, Chulalongkorn Publishers, Bangkok, Thailand, 1997.



Haemwaan Sivaraks



Athasit Surarerks