A Comparison of Vector Quantization Codebook Generation Algorithms Applied to Automatic Face Recognition.

C. S. Ramsay[†] K. Sutherland[†] D. Renshaw and P.B. Denyer
Integrated Systems Group, Electrical Engineering Department,
University of Edinburgh, Edinburgh, EH9 3JL.

Abstract

Automatic facial recognition is an attractive solution to the problem of computerised personal identification. In order to facilitate a cost effective solution, high levels of data reduction are required when storing the facial information. Vector Quantization has previously been used as a data reduction technique for the encoding of facial images.

This paper identifies the fundamental importance of the vector quantizer codebooks in the performance of the system. Two different algorithms – the Linde-Buzo-Gray algorithm and Kohonen's Self Organising Feature Map – have been used to obtain two sets of facial feature codebooks. For comparison, the system performance has also been analysed using a codebook dedicated to the test population. It has been shown that by using a *good* codebook generation algorithm it is possible to substantially reduce the dimensionality of the vector codebooks, with remarkably little degradation in system performance.

1 Introduction

Automatic facial recognition is now receiving an increasing amount of research interest [1], largely due to its possible applicability to the automatic personal identification task. As such, automatic face recognition is attempting to stake its claim among the other biometric systems available (notably fingerprint, hand geometry, dynamic signature matching and voice pattern recognition).

In order to establish biometric systems as likely tools for personal identification we must instill public confidence in the concept of biometric storage and retrieval. In this area automatic face recognition scores heavily over the other likely biometrics, as the least intrusive and the most *natural* personal identification process. It is also important that any prototype system has a demonstrably high accuracy. To obtain such high accuracy a likely advancement would be to incorporate the use of a number of different biometrics into one identification system, making the final response conditional on all the available information.

However, in practice, the overriding factor in achieving a *marketable* system will be the data reduction obtained, since this controls the speed at which multiple comparisons can be performed and, ultimately, the cost, as data storage

[†]These authors have SERC studentships.

requirement is a prime factor in this area. The available level of data reduction is now of particular importance given the increasing use of smart-card technology and the ever increasing likelihood that we will all eventually carry personal biometric information with us in this way. If a facial image is to be one of many pieces of biometric information stored on our smart-card, then accurate, high data reduction, parametrisation of the face is required.

The authors have previously introduced a Vector Quantization (VQ) based facial recognition technique and presented results for a recognition experiment using 30 individuals [2]. In this paper we would like, firstly, to present further results on a 33% larger data set and, secondly, to investigate the use of VQ codebook generation techniques to reduce further the data space required to store the intrinsics of the face (or the ***facial signature***). The codebook generation techniques used here to reduce the overall number of vectors are Kohonen's neural Self-Organising Feature Map and the Linde-Buzo-Gray algorithm. However, firstly, a brief overview of the entire facial recognition algorithm will be presented.

2 System Overview

The intrinsic function of a pattern recognition process is that of parameterisation, *i.e.* the extraction of the fundamental characteristics of the object to be recognised. In [2] the authors presented an algorithm which distilled the face into a compact facial signature, while still maintaining much of the ***recognisability*** of the initial input face.

In brief, the system identifies seven fundamental features; each eye, the nostrils, the bridge of the nose, the mouth, the chin and the hair. In addition to storing these individual facial parts, the facial signature also incorporates a coarsely sub-sampled view of the entire face. The partitioning of the face used here is shown in Figure 1. These facial features are automatically located using a template matching algorithm based on Fischler and Elschalger's *feature embedding* approach [3]. Manual correction of the location failures was performed.

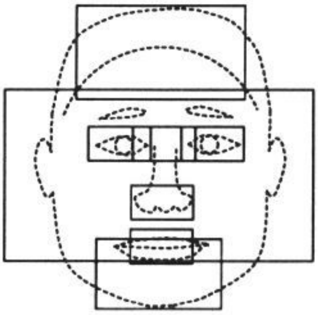


Figure 1: Facial Features Exploited to Differentiate between Individuals.

Having extracted these eight facial parts for storage and discarded the rest of the image data, there still remains the task of data reduction. The signal processing function of VQ has been chosen to provide the requisite reduction. The use of VQ for image coding is widespread [4]. However, recognition analysis using VQ is much more novel.

Applied to faces, VQ performs a function analogous to the police *photofit* system. In turn, each of the eight selected features is compared with a codebook of standard examples (or *vectors*) of only that feature; using the normalised euclidean distance as a metric, the most similar of these standard examples is chosen as the most likely match. Then to store that feature, we need only to keep the *index* of the standard feature and not all the data points which constitute that vector.

It is desirable to train a facial recognition device on a number of examples of each person it is expected to recognise, in order to allow it to construct internal representations of each person. To facilitate this, a system of *feature histograms* has been devised; these reflect the ways in which the subject's face has varied during training. To obtain the histograms we use the VQ technique described above to encode each feature, from each training image, then that *match* is recorded in the histogram. Thus, given 20 possible vectors and ten training images one particular *histogram* could look like this, Figure 2.

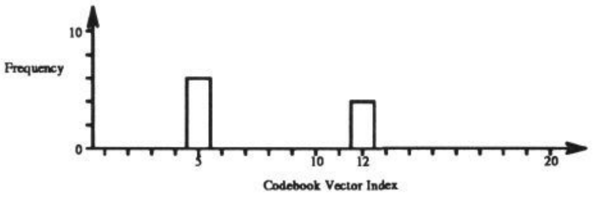
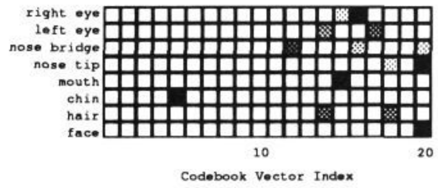


Figure 2: One Feature Histogram for One Person's Training Images. This feature has been mapped six times to codeword 5 and four times to codeword 12 during training.

In order to differentiate one person from another their respective histograms would have to show significant differences. If we look at the histograms of all eight features for two different people, Figure 3, it can be seen that there is only a low level of variability within training for each person, yet there is a substantial difference between these characteristics between these two people. Low *within person* and high *between person* variabilities are essential for a good recognition system.

By storing the histograms obtained in the manner described above it is argued that we have the requisite facial data to perform recognition. Thus, the feature histograms can be thought of as our facial signatures. The problem with this approach is that, to date, the VQ codebook entries have been extracted from a control image of each member of the test population. In this way the data storage requirement, as determined by the number of vectors, is dependent on the population size. Thus, if we were to enlarge the population there would be a consequent rise in data storage requirements. To combat this, the following section outlines the ways in which the codebook dimensionality can be reduced.



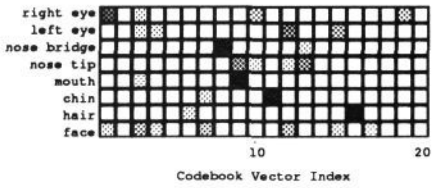


Figure 3: Feature Histograms for Two Population Members.
In this graphic all eight feature histograms are shown, with grey tone signifying the frequency of use of each codebook vector. *Black* represents a well used vector, with lighter tones indicating less use.

3 Codebook Generation

The process of VQ relies on having a good set of vectors in its codebook. For image compression, these vectors are chosen to minimise the overall pixel level error introduced. However, for face recognition, other considerations are also important. For example, distinctiveness may be more significant than pixel error when encoding facial features. Fortunately, there are many different algorithms available which perform codebook generation [5].

As an initial starting point, each feature codebook has been constructed using one sample vector drawn from a control image of each member of the test population, thus reflecting only the variation present within this test population. To obtain a reduction in data storage, the two most common VQ codebook generation techniques have been used to reduce the codebook dimensionality. In this study the test population contained 40 members and thus the initial codebooks (for each of the eight features) contained 40 vectors. A reduction to half this number has been chosen as a sufficiently demanding test of the codebook generation algorithms available. The task of the codebook generation technique is to perform the requisite dimensionality reduction while still containing the system's error rate within acceptable limits.

3.1 Kohonen's Self-Organising Feature Maps

Neural network clustering techniques have been employed in a variety of application areas, such as pattern recognition, optimisation and, notably, VQ codebook design for image compression [6, 7]. Kohonen developed his Self-Organising Feature Maps (KSOFM) in order to model the neural feature maps which are thought to form in the human brain [8]. KSOFMs result in a network where neighbouring output units have similar responses : ***topological ordering*** has occurred. This ordering can be used to reduce search requirements in VQ applications, though this is not an aspect of the KSOFM which has been exploited here. As a clustering algorithm, KSOFM allows a reduction in the dimensionality of the input vector space to a smaller number of reference vectors.

KSOFM defines a matrix (usually two dimensional) of output units, or neurons, each of which has a weight vector, w_j, associated with it. Input vectors from a training set are presented sequentially and the weight vectors are adjusted as described below. The weight vectors converge towards cluster centres after sufficient training time. The reason that topological ordering occurs is that each input vector adjusts not only one weight vector, but a *neighbourhood* of weight vectors.

The KSOFM algorithm is defined thus :

Step 1. Initialise all weight vectors to random values.

Step 2. Apply new input vector, $x(t)$.

Step 3. Calculate the Euclidean distance from $x(t)$ to all output nodes j

$$d_j = \sum_{i=0}^{N-1} (x_i(t) - w_{ij}(t))^2$$

where N is the dimensionality of the input vector.

Step 4. Select the output node with the smallest distance d_i and label it as the winning unit, j^*.

Step 5. Update weight vectors of all nodes in the matrix according to the equation

$$w_{ij}(t+1) = w_{ij}(t) + \eta(t, \mathcal{D})\alpha(t)(x_i(t) - w_{ij}(t))$$

where $\eta(t, \mathcal{D})$ is the neighbourhood gain function, and $\alpha(t)$ is the adaption gain function.

$\eta(t, \mathcal{D})$ is a function which defines a neighbourhood on the matrix round the winning neuron, decreasing exponentially with distance $\mathcal{D}$ from the winning neuron, and which shrinks over time. $\alpha(t)$ decreases exponentially with time, and $0 \leq \alpha(0) \leq 1$.

Step 6. Repeat **Step 2** to **Step 5** until the entire training set has been presented e times. e is the number of *epochs*, which is set before training starts.

In this application the input vectors, x, come from the original feature codebooks containing 40 vectors, and the matrix was a 5×4 array of neurons producing a codebook of 20 vectors.

3.2 Linde-Buzo-Gray Algorithm

The Linde-Buzo-Gray (LBG) algorithm [9] is the most commonly used codebook design algorithm due to the fact that it was the earliest proposed method and consistently outperforms other methods in a variety of applications [10, 11]. It is an iterative technique which repeatedly moves codewords to cluster centroids in an effort to find a codebook which will display the lowest error when encoding the training data (*i.e.* a modified version of the K-Means clustering algorithm). The basic algorithm is as follows :

Step 1. Initialise the codebook.

Step 2. For each vector, x, in the training set, calculate the Euclidean distance from it to each vector v_j in the codebook.

$$d_j = \sum_{i=0}^{N-1} (x_i - v_{ij})^2 \tag{3}$$

where N is the dimensionality of the vectors.
The minimum distance selects the closest vector, v_j^*, in the codebook. Assign x to the cluster around v_j^*.

Step 3. Replace each codeword with the centroid of the vectors in the training set that have been assigned to it. If any codewords are unused, they are discarded and replaced with new codewords which are more likely to be used in the next iteration. In this work, this has been done by taking the most commonly used codewords and splitting them to create two close copies of the original.

Step 4. If the total error in clustering the training data is still decreasing by a significant amount, return to **Step 2.** Otherwise, stop.

This algorithm should optimise the codebook so that the sum of the distances from each vector of the training set to its nearest codeword is a minimum. It is possible, however, that in certain conditions the algorithm will reach a local minimum rather than a global minimum. This can be seen to be true due to the fact that the final codebook will change if the initial codebook is different.

There are a number of recognised methods for generating an initial codebook, the most common of which is to populate the codebook with vectors chosen randomly from the training set. Another method is to use a codebook generated from other clustering techniques, such as the KSOFM technique described above, as the initial codebook. The method utilised here was to populate the codebook with 20 replicas of the centroid of the entire training set. In this way the codebook is gradually filled with useful codewords, as unused codewords are replaced in **Step 3**.

There are some similarities between the ways in which the two techniques outlined above perform dimensionality reduction. However, the codebooks produced can be significantly different. To illustrate this Figures 4 and 5 show the two 20 vector *face*[†] codebooks generated from the same original input of 40 vectors. It can be clearly seen that both codebooks contain composite faces formed by the merging of several of the input faces together, but that they are quite different from each other.

[†]There are parallel codebooks for each of the other seven features used in the recognition algorithm.

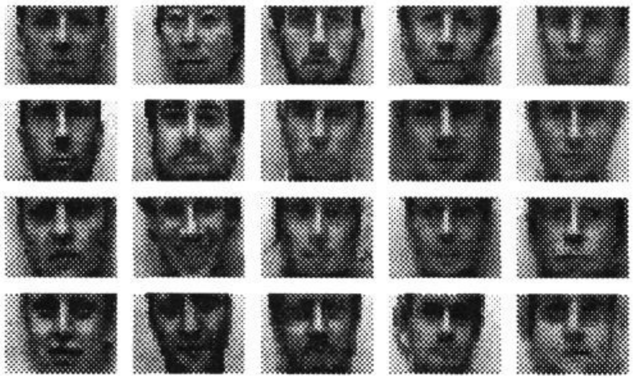


Figure 4: Codebook Generated using LBG.

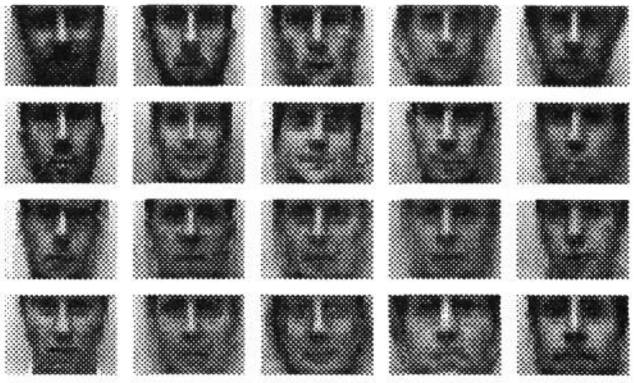


Figure 5: Codebook Generated using KSOFM.

4 Experimental Results

At present the algorithm for facial recognition is implemented as a suite of software programs. The system cannot yet function in real-time and thus the test images used are stored on disk for repetitive analysis. The images used were captured with a video camera under largely controlled conditions. However, the subjects were allowed to vary their expressions during the several days during which images were being captured. In this way, the results obtained for the system will be closer to a real-world implementation than some other, highly controlled, studies.

As mentioned earlier, a test population of 40 males was used, with ten images of each person used for training (*i.e.* the generation of the facial signature) and ten images kept for testing. The trial thus consisted of 400 test presentations. For each presentation, the output ranking of likelihood was recorded. From this data, and from knowing the correct response, it is possible to obtain a first place recognition rate (*ie* the proportion of tests in which the correct signature was selected as the most similar to the test stimulus). By analysing the output ordering of responses an average rank figure can also be obtained.

4.1 Dedicated codebook

To provide a benchmark, and to demonstrate the best possible performance, a set of dedicated codebooks were used. These codebooks were constructed using the same vectors as were used to train the other codebook generation algorithms. The feature histograms were thus 40 vectors wide for each of the eight features. For this experiment the average rank and the cumulative success rates of the first three output positions is given in Table 1.

Average Rank	1.37
1^{st} Place Recognition	88.5%
2^{nd} Place Recognition	92.75%
3^{rd} Place Recognition	95.5%

Table 1: Recognition rates for a 40 member population with 400 test presentations.

4.2 Dimensionality Reduction

Essentially, here, we are performing the same experiment as above, but using codebooks of half the size. Such a significant reduction would be expected to cause a substantial reduction in recognition performance unless, as described in section 3, the algorithms used to derive the new codebook entries reflect the initial characteristics of all 40 vectors. The experiment is thus a parallel comparison between the two different sets of codebook vectors when applied to the recognition function. Table 2 gives the relative performances between the two approaches.

	KSOFM	LBG
Average Rank	1.99	1.46
1^{st} Place Recognition	70.2%	83.3%
2^{nd} Place Recognition	81.5%	90.0%
3^{rd} Place Recognition	88.5%	94.7%

Table 2: Two methods of codebook design.

4.3 Discussion

If we consider the first experiment using a dedicated codebook, the performance results are very encouraging. The first place recognition rate of 88.5% does not yet rival the other biometric systems. However, the results reported here represent one of the very few significant population studies of a practical automatic face recognition system reported to date and, as such, give an important indication of the future viability of automatic face recognition.

Considering the dramatic reduction in codebook dimensionality, the second set of recognition results are remarkably good. The LBG approach performs significantly better than the KSOFM method. Its overall recognition rate is approaching that of the 40 vector system reported in section 4.1. To explain the variation in the performance of these two codebook generation algorithms more detailed consideration of their mechanics is required.

In general the codebook generation algorithms perform reduction by averaging the most similar vectors together, while still maintaining good coverage of the initial vector space. If the most similar vectors are also the most frequently used vectors, then the clustering process may reduce the good spread of vector choice required to maintain good differentiation when encoding the population. To investigate whether this is in fact the case feature histograms have been drawn up for the entire population.

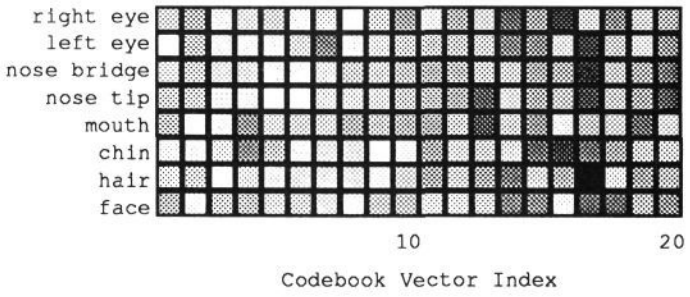


Figure 6: Population Feature Histograms for LBG.

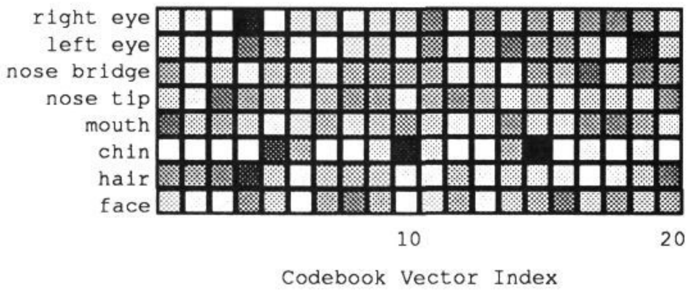


Figure 7: Population Feature Histograms for KSOFM.

Figures 6 and 7 illustrate the feature histograms for the LBG and KSOFM algorithms respectively. The spread of vector choice is much more even for the LBG vector set. We believe this is the underlying factor which explains the relatively poor performance of the KSOFM approach. In effect, KSOFM has maintained *too good* coverage of the entire feature space at the expense of differentiation between the most common feature types.

5 Conclusions and Future Work

The results of the initial research into the use of VQ for automatic facial recognition are encouraging. This approach has been shown to work for facial recognition and undoubtedly has uses in other areas of image pattern recognition.

The dimensionality of the codebooks used by VQ based recognition has been identified as a potential limiting factor, however, this research has shown the important improvements which can be obtained in this area by using different codebook generation algorithms. However, we accept that further experimentation, with different populations, is required to validate the results reported here. It is further recommended that much larger populations are required to adequately test potential facial recognition systems.

References

[1] V Bruce and M Burton. Computer recogniton of faces. In A W Young and H D Ellis, editors, *Handbook of Research of Face Processing*, pages 487–506. North-Holland, 1989.

[2] K Sutherland, D Renshaw, and P B Denyer. A novel automatic face recognition algorithm employing vector quantization. In *Digest of the IEE Colloquium on Facial Recognition and Storage*, London, January 1992.

[3] M A Fischler and R A Elschalger. The representation and matching of pictorial structures. *IEEE Transactions on Computers*, 22:67–92, 1973.

[4] N M Nasrabadi and R A King. Image coding using vector quantization : A review. *IEEE Trans. on Comms.*, COM-36(8):957–971, August 1988.

[5] R M Gray. Vector quantization. *IEEE ASSP Mag.*, 1(2):4–29, April 1984.

[6] N M Nasrabadi and Y Feng. Vector quantization of images based upon the Kohonen self-organising feature maps. In *Proc. Int. Joint Conf. on Neural Networks (IJCNN-88)*, pages 101–108, July 1988.

[7] T-C Lee and A M Peterson. Adaptive vector quantization using a self-development neural network. *IEEE J. on Selec. Areas in Commun.*, 8(8):1458–1471, October 1990.

[8] T Kohonen. Clustering, taxonomy, and topological maps of patterns. In *Proc. 6th Int. Conf. on Pattern Recognition*, pages 114–128. IEEE, October 1982.

[9] J Linde, A Buzo, and R M Gray. An algorithm for vector quantizer design. *IEEE Trans. on Comms.*, COM-28(1):84–95, January 1980.

[10] J D McAuliffe, L E Atlas, and C Rivera. A comparison of the LBG algorithm and Kohonen neural network paradigm for image vector quantization. In *Proc. Int. Conf. on Acoustics, Speech and Signal Processing (ICASSP-90)*, pages 2293–2296. IEEE, April 1990.

[11] W Equitz. Fast algorithms for vector quantization picture coding. In *Proc. Int. Conf. on Acoustics, Speech and Signal Processing (ICASSP-87)*, pages 725–728. IEEE, April 1987.