Improvements in HMM-based isolated word recognition system

A.M. Peinado
J.M. Lopez
V.E. Sanchez
J.C. Segura
A.J. Rubio Ayuso

Indexing terms: Speech recognition, Mathematical techniques

Abstract: A speaker-independent isolated word recognition system for voice-activated robots is introduced. Since such an application requires a fast and accurate recognition system, discrete hidden Markov models and vector quantisation techniques have been used. Different initialisations for VQ and HMM training are tested to obtain improvements over the other systems. The final models can be evaluated using a temporal normalisation of the HMM scores. A threshold-based rejector can also be established using this temporal normalisation.

1 Introduction

A speaker-independent isolated word recognition system for use on a voice-activated robot is presented. This application requires a fast and accurate recognition system, so the well known discrete HMM technique, along with VQ, is used. This method requires a sophisticated training, but performs very efficiently in the recognition task when compared with other well known methods [1].

A traditional problem in engineering is to obtain a model for a specific process. Models are useful in problems like prediction, recognition or identification. A HMM probabilistically models the generation of a given word, so as to recognise it. A HMM is a generator that produces observation sequences according to a certain probability distributions. These observations can belong to a discrete or continuous vectorial space (discrete or continuous HMMs). The continuous models are more accurate in recognition than the discrete models. The training is more complicated and the recognition requires more computation for continuous models. Since the vocabulary is relatively short, the digits and six command-words) discrete HMMs are preferred. The development of a HMM speech recognition system is covered in detail in References 2–4.

The speech signal has to be coded as an observation sequence for the discrete HMM. This coding is developed by a vector quantisation (VQ) process. VQ is a well known technique for bit rate reduction. VQ can be visualised as a two-step process as shown in Fig. 1 in the spectral sense. The first step, called identification, consists of finding a spectral model F' which best models the input speech frame spectrum F_x. LPC analysis [5–7] is applied at this step. The second step is quantisation in which another spectral model F, belonging to a finite set of spectral models (called a codebook), approaches F' (F can be used for transmission or storage). F is chosen to minimise a defined spectral distortion measure $d(F_x, F)$, or equivalently $d(F', F)$ [8]. The development of VQ technique is covered in detail in References 9–11.

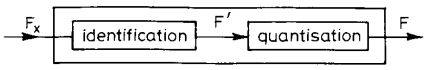

Fig. 1 *Block diagram of quantiser*

The HMM and VQ training algorithms both converge to a local optimum point. This convergence is very dependent on the initialisation of these algorithms. Different initialisation methods are proposed and the convergence and results studied. The performance of two input sequence evaluation methods, forward-backward and Viterbi, is studied for HMM recognition. A temporal normalisation of the input sequence evaluation score is introduced to obtain a model probability independent of the duration of the spoken word. Once this normalisation is established, it is possible to compare the probabilities corresponding to several utterances and to fix a probability threshold for utterance rejection. Temporal normalisation allows information to be obtained about the quality of the models being constructed and the recognition performance.

VQ and related issues, type of feature vectors, distance measures, training and initialisations, are first treated. HMM issues, training and recognition, are then presented. The problem of the initial models is posed, and the temporal normalisation and the threshold-based rejection are treated. Finally the parameters and conditions of the system and the results of VQ, recognition, duration inclusion and rejection are described, and a summary is given.

2 Vector quantisation

VQ is a data compression method, useful for data transmission or storage [9, 10]. Each input frame vector, made up with a set of features obtained in the analysis process, is represented by a single symbol. This property allows the discrete HMM to be used, since they are fed with symbols from a finite codebook.

Paper 7984I (E7, E8), first received 25th October 1989 and in revised form 14th January 1991

The authors are with the Departmento de Electronica y Tecnologia de Computadores, Universidad de Granada, Campus Fuentenueva s/n, 18071 Granada, Spain

In a pure vectorial sense, the VQ consists of finding the nearest neighbour of an input frame vector, among the centroids (codewords) of a codebook [12]. The input vector can be represented only by the centroid index, once the codebook is known. Suppose that the representation space is partitioned into cells S_j ($j = 1, \ldots, N$). Each cell has a centroid $\boldsymbol{c}_j$. When an input vector $\boldsymbol{c}$ arrives, it has to be classified in a cell

$$\boldsymbol{c} \in S_j \Leftrightarrow d(\boldsymbol{c}, \boldsymbol{c}_j) \leqslant d(\boldsymbol{c}, \boldsymbol{c}_i) \quad i = 1, \ldots, N \quad i \neq j$$

where d is a defined distance in the vectorial space V, $d: V \times V \rightarrow R$ (the number of dimensions of V is the number of features). Then, the input frame vector $\boldsymbol{c}$ is coded by the nearest neighbour centroid index j.

The next problem, that has been widely studied, is the choice of the distance measure. The distance measure that best matches two acoustic events for the features used is chosen. A weighted cepstral distance measure and an Euclidean distance measure with liftered cepstrum have been proposed and successfully applied in speech recognition, but when different features are included in the feature vectors, a multifeature weighted distance measure (MWDM) is preferred [13]. The MWDM is given by

$$d_{MW} = \frac{\sum_{i=1}^{Q} (\tilde{c}_t(i) - \tilde{c}_r(i))^2 + \sum_{i=1}^{Q} (\Delta\tilde{c}_t(i) - \Delta\tilde{c}_r(i))^2}{\sigma_c^2} + \sum_{I=1}^{L} \frac{(F_t^{(I)} - F_r^{(I)})^2}{\sigma_I^2} \quad (1)$$

where $\tilde{c}_t(i)$ and $\Delta\tilde{c}_t(i)$ are the liftered cepstral and delta cepstral coefficients. Index t denotes the test frame and r the reference frame, Q is the number of cepstral coefficients. $F_t^{(I)}$ and $\sigma_{(I)}^2$ denote other kinds of features (energy and delta energy, in this case) and their variances. L is the number of new features and σ_c^2 is the variance of the cepstral terms.

A full search optimal vector quantiser that looks for the nearest neighbour centroid in all the codebook has been developed, using eqn. 1 with a feature vector composed of the cepstral coefficients, the delta cepstral coefficients, the frame energy and the frame delta energy. There also exists a fast suboptimal solution using a tree search that is not considered in this work.

2.1 Codebook implementation

The main problem in VQ is the codebook generation. The widely applied iterative algorithm for clustering called k-means has been used for this purpose. Getting a suitable codebook is very important to obtain a good characterisation of the frames and, therefore, a good recognition rate.

Consider a training set $\{\boldsymbol{c}\}$ of vectors of a vectorial space V, and a distance $d: V \times V \rightarrow R$. It is necessary to assign each vector to one of the k clusters. The algorithm consists on the following steps:

(*a*) Initialisation of k cluster centroids $\boldsymbol{c}_1(0), \ldots, \boldsymbol{c}_k(0)$.

(*b*) At the lth iteration, distribute the vectors $\{\boldsymbol{c}\}$ among the k centroids

$$\boldsymbol{c} \in S_j(l) \quad \text{if } d(\boldsymbol{c}, \boldsymbol{c}_j(l)) \leqslant d(\boldsymbol{c}, \boldsymbol{c}_i(l)) \quad i \neq j$$

where $S_j(l)$ denotes the set of vectors whose nearest neighbour centroid is $\boldsymbol{c}_j(l)$.

(*c*) It is necessary to compute the new centroids from these results. The centroid of $S_j(l+1)$ is that which minimises the total distortion cell

$$D_j = \sum_{\boldsymbol{c} \in S_j} d(\boldsymbol{c}, \boldsymbol{c}_j(l+1))$$

The vector $\boldsymbol{c}_j(l+1)$ that minimises D_j is the mean vector of $S_j(l)$. This is correct for the Euclidean distance and for the MWDM, but, in general, $\boldsymbol{c}_j(l+1)$ must be found.

(*d*) If the new centroids match the previous ones the algorithm has converged and the procedure ends. Otherwise, go to step *b*.

The behaviour of the k-means algorithm is influenced by the parameter k, and the initialisation and the geometrical properties of the vector set $\{\boldsymbol{c}\}$ [14]. So when an arbitrary initialisation is used, a bad and slow convergence can be expected. For this reason two alternative methods of initialisation are introduced.

2.2 Initialising the k-means algorithm

2.2.1 k-means type initialisation: The procedure consists of applying the k-means algorithm to a single training sequence $\{\boldsymbol{c}^t\}$. Each sequence consists of several utterances of each word of the vocabulary from the same speaker. An arbitrary set of vectors is used as the initial codebook, $\boldsymbol{c}_1^r = \boldsymbol{c}_{1 \cdot p}^t$, $\boldsymbol{c}_2^r = \boldsymbol{c}_{2 \cdot p}^t$, $\boldsymbol{c}_3^r = \boldsymbol{c}_{3 \cdot p}^t$, ..., $\boldsymbol{c}_k^r = \boldsymbol{c}_{k \cdot p}^t$, where $p = \text{ent}(N/k)$, N is the number of training vectors and k is the codebook size.

After this step, it is hoped that the algorithm converges easily, because of the low density of training vectors in the space. A meaningful initial codebook in which all the words are represented is thus obtained.

2.2.2 Splitting initialisation: Unlike the above method, all the training sequences are used for initialisation. The procedure consists of four steps [9]

(*a*) The absolute training set centroid has to be computed. This is an initial codebook $\{\boldsymbol{c}_1\}$ with $N_1 = 1$ centroids.

(*b*) At the lth step, the number of centroids of the codebook is duplicated by perturbing the first coefficient of each centroid, and a new codebook $\{\boldsymbol{c}_i, i = 1, \ldots, N_{l+1}\}$, with $N_{l+1} = 2N_l$, is obtained.

(*c*) Apply the k-means algorithm to the codebook obtained in the previous step.

(*d*) If N_{l+1} is the required number of centroids, the process ends. Otherwise, go to step *b* for a new iteration.

3 Hidden Markov models

3.1 Concept overview

A HMM is a model of speech generation, although it is used for recognition purposes. A HMM is characterised by five elements [4]

(i) A state set S with N states $S = \{S_1, \ldots, S_N\}$. Denote the current state at time t as q_t.

(ii) A symbol set V with M different symbols $V = \{V_1, \ldots, V_M\}$. Denote the current symbol at time t as O_t.

(iii) A transition probability distribution $A = \{a_{ij}\}$, where a_{ij} is the transition probability from state s_i to state s_j.

(iv) A symbol probability distribution $B = \{b_j(k)\}$, where $b_j(k)$ is the production probability of symbol v_k at state s_j.

(v) An initial state distribution $\Pi = \{\pi_i\}$, where π_i is the probability that the symbol sequence generation begins at state s_i.

In Fig. 2 it can be seen how a HMM generates a symbol sequence $O = O_1, \ldots, O_T$. This process also generates a

state sequence $Q = q_1, \ldots, q_T$. This state sequence is hidden (from which the adjective hidden comes). Fundamentals of HMMs can be seen in References 2–4 and 15.

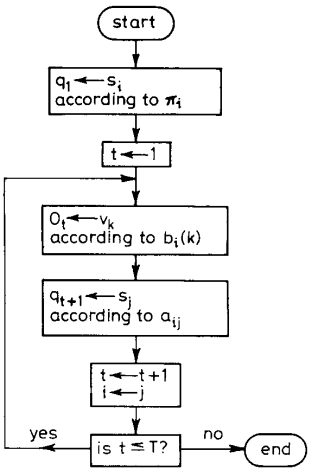

Fig. 2 *Flow chart of generation of sequence $O = O_1, O_2, O_T$ by HMM*

3.2 Speech recognition by HMM

For a vocabulary with L words $\{V_1, \ldots, V_L\}$, the recognition system is a set with L HMMs $\{\lambda_1, \ldots, \lambda_L\}$, where each model λ_i is associated to one word V_i of the vocabulary. Given an input symbol sequence, the generation probability $p(O \mid \lambda_i)$ is computed for every model λ_i. The model giving the highest probability corresponds to the recognised word. More than one model per word can be used.

For speech recognition, it is suitable to use the left-to-right model, because it matches the sequential nature of the speech generation process. A scheme of a five-state left-to-right model is shown in Fig. 3. A structure in

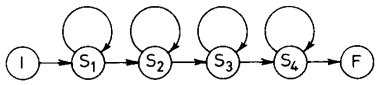

Fig. 3 *Left-to-right model*

which an initial and a final state (I and F) are added is used. They have no production probabilities. These models are defined as follows:

$$\pi_i = \begin{cases} 1 & i = I \\ 0 & i \neq I \end{cases} \quad a_{ij} = 0 \quad (j < i, j > i + \Delta)$$

Note $\Delta = 1$ in this work.

The recognition process fundamentally consists of computing the $p(O \mid \lambda)$ probabilities. The general procedure of computation is

$$p(O \mid \lambda) = \sum_{all\ Q} \pi_{q_1} b_{q_1}(O_1) a_{q_1 q_2} b_{q_2}(O_2) \cdots b_{q_{T-1}}(O_{T-1}) a_{q_{T-1} q_T} b_{q_T}(O_T) \quad (2)$$

But eqn. 2 requires a very high number of computations. Both the forward–backward and the Viterbi algorithms have been used because they are much more efficient for this task. The Viterbi algorithm provides an approximate solution. Both algorithms are treated in detail in References 3 and 4.

3.3 HMM training

The training of the models is the more complicated problem relating to HMMs. Several solutions have been proposed. Rabiner *et al.* [4] proposed an algorithm called the Baum–Welch re-estimation algorithm. Boite *et al.* [16] used an iterative procedure for model training based on the Viterbi algorithm or Viterbi re-estimation.

The Baum–Welch algorithm is the optimal solution to this problem, since it is based on a standard Lagrange maximum search. Experience indicates that when the Baum–Welch algorithm is used the resulting models are very dependent on the initial models. This dependence is not so strong when a Viterbi re-estimation is used. Taking into account the above points, the following procedure has been used for training:

(*a*) Some initial models are computed by a linear or manual segmentation of training sequences.

(*b*) A Viterbi re-estimation is performed up to convergence. In each iteration, the segmentation properties of the Viterbi algorithm are used for computing the new model parameters as occurrence frequencies.

(*c*) The Baum–Welch algorithm is applied to the models obtained at step *b*. In each iteration, the re-estimation formulas [4] are applied for computing the new model. As result, the models obtained at step *b* are optimised.

These algorithms keep the probability of the events that are not represented in the training data equal to zero. This can be considered an effect of an insufficient amount of training data, and it can be avoided by setting these probabilities equal to a certain small value ε.

4 Initial models

Both Viterbi and Baum–Welch re-estimations are very dependent on the initial models. They only guarantee convergence to a local maximum of likelihood. Several methods of initialisation must be posed before the re-estimation to look for a better local likelihood maximum. Two initialisation methods have been tested based on the segmentation of the training utterances, in which segments are associated with states.

From a segmentation, the transition and the production probabilities can be computed as occurrence frequencies just by dividing the number of occurrences of a partial event (part of a certain global event) by the number of occurrences of the global event. These global events are the transitions from a certain state to any other state, or the production of any symbol in a certain state.

A test with two alternative segmentations has been carried out: a manual and a linear segmentation:

(i) The manual segmentation is performed over a few sequences of the training set. In this case, segments are approximately associated to phonemes, so as many states as phonemes in the word are obtained. A certain degree of cohesion in the states is thus guaranteed. An appropriate number of states for each model is also achieved.

(ii) With the linear segmentation, each training utterance is divided into a fixed number of states, with equal length for all the segments in a given training utterance.

In the case of manual segmentation, a different insufficient training method is used in the initialisation step. The problem is that when manual segmentation is used, a smaller number of training sequences is taken for initialisation, and symbols that do not appear in the initialisation sequences could appear in other training ones.

For symbols that do not appear in the training sequences, they are considered to have a probability F times smaller than that of the least probable symbol that appears in the training data. F is called the acceptance factor.

5 Score temporal normalisation

It would be interesting to be able to match probabilities from different sequences. The probability $p(O \mid \lambda)$ strongly depends on the sequence duration T. That is the reason why it is necessary to do a temporal normalisation of the log-score $\log p(O \mid \lambda)$.

In any of the two proposed algorithms, $p(O \mid \lambda)$ is computed as addition of products of $2T$ probabilities

$$\pi_{q_1} \prod_{s=1}^{T-1} a_{q_s q_{s+1}} \prod_{s=1}^{T} b_{q_s}(O_s)$$

The temporal normalisation consists of computing an average probability (geometric average)

$$\overline{p(O \mid \lambda)} = (p(O \mid \lambda))^{1/2T} \tag{3}$$

The meaning of the normalised score is a mean probability per symbol of the utterance. Applying logarithms to eqn. 3, a mean normalised log-score is obtained

$$\log \overline{p(O \mid \lambda)} = \frac{\log p(O \mid \lambda)}{2T} \tag{4}$$

The factor of 2 can be ignored in eqn. 4.

6 Including duration information: threshold-based rejection

The main problems in recognition are similar to those of training. In recognition, it is usual to use some additional information such as energy and duration, in a postprocessing stage. Energy information is already used in the feature vectors, so the postprocessing is developed only with duration.

State duration and word duration can be included in the postprocessing as two new scores, P_{sd} and P_{wd}, respectively, where

$$P_{sd} = \sum_{i=1}^{N} \log (p_i(d_i)) \tag{5}$$

$$P_{wd} = \log (p_w(T)) \tag{6}$$

where $p_i(d_i)$ is the duration distribution of state i, T is the utterance duration, and $p_w(T)$ is a Gaussian distribution of word duration. Three ways of calculating the state duration distribution are considered:

(*a*) Histograms (SD1)

(*b*) Histograms with normalised duration $d_i \cdot \bar{T}/T$ ($\bar{T}$ is the mean word duration) (SD2)

(*c*) Gaussian distributions with normalised duration (SD3)

Word duration is easily modelled by a Gaussian density, considering that the word duration process is a Gaussian process (what is basically true). P_{sd} and P_{wd} are incorporated to the word log-score using experimental weights.

It is also possible to incorporate the state duration densities into the HMM algorithms (forwardbackward, Viterbi and Baum–Welch formulas). This considerably increases the amount of computation [4]. It is preferable to include the duration information in a postprocessing.

In the postprocessing stage, one possibility to diminish the error rate is to reject those utterances that are not clearly recognised. The best way to include temporal information in a threshold-based rejection technique is sought in a later section.

The rejection method consists of defining a score threshold for each HMM of the vocabulary. This is possible because of the temporal normalisation, explained above, that extracts the temporal dependence of the score. This threshold is about $\bar{x} - \alpha\sigma_x$, where x is the log-score obtained from a given HMM, and $\bar{x}$ and σ_x are the log-score mean and the log-score standard deviation, obtained from the training data. Moving this threshold, by the factor α, it is possible to obtain several rejection percentages. Several experiments were performed to find the best rejection.

7 Experimental results

7.1 System parameters and conditions

The data were sampled at 8.091 KHz, and pre-emphasized with a pre-emphasis factor $\mu = 0.95$. Hamming windows were applied to blocks of 256 samples, with an overlapping of 64 samples. Liftered cesptrum was computed for each frame (with $Q = 10$ and length 12 for the liftering window) and delta cepstrum was approximated by linear regression on a ± 3 frame environment. Frame energy was normalised to the peak of energy in the word and expressed in the decibel scale. Delta energy was computed from the normalised decibel-scaled values of energy. An average of all of these parameters was performed at every other consecutive frame. The final result is as if 256 sample frames overlapped 128 samples.

All the utterances were isolated by an explicit endpoint detector [17], which was modified in Reference 12. These utterances were coded with a 64-centroid codebook in all the experiences. One model was used per word, and when linear segmentation is used for HMM initialisation there were seven states per model.

The vocabulary consists of the ten Spanish digits and the Spanish words {cuerpo, hombro, codo, muñeca, mano, dedos}, for controlling each motor of a robot.

The database consists of 40 speakers and three utterances per speaker per word (1920 words). It was recorded under the normal conditions of work rooms, so certain level of noise, such as computer noise, is included. The conditions of recording (echo and noise conditions) were variable along the time, from the first speaker up to the last speaker. Two subsets of this database were considered for the experiments:

(*a*) the first 20 speakers (DB1) for training

(*b*) the last 20 speakers (DB2) for testing.

With this choice, the error rate is not near 0%, because of the variable conditions of the recording. The variations of error rates and rejections in our experiments can be observed, and a real situation of environment change of the recognition system can be simulated.

7.2 k-means convergence

The convergence curves for the k-means algorithm are shown in Fig. 4, using DB1. The convergence curve for k-means type initialisation is represented by the continuous line. The dotted line represents the splitting initialisation. The total distortion D against the iteration

number is shown, where

$$D = \sum_{i=1}^{64} \sum_{c \in S_i} d(c, c_i)$$

and where S_i is the cluster corresponding to the centroid c_i.

No important differences between the two convergences can be observed from Fig. 4.

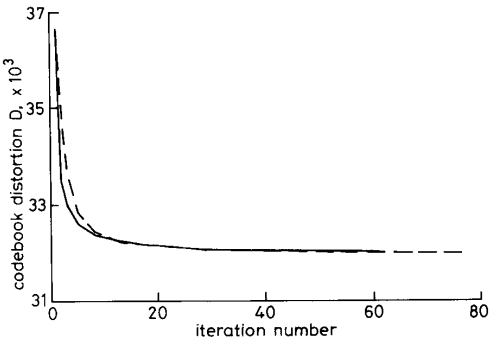

Fig. 4 *Total distortion against iteration number of k-means convergence*
- - - - Splitting initialisation
——— *k*-means initialisation

7.3 Recognition results

The error rates, using splitting and *k*-means type initialisations for the *k*-means algorithm and manual and linear segmentations for HMM training, are shown in Table 1, for both Viterbi and forward–backward score evaluation algorithms. DB1 and DB2 are used for training and testing, respectively. In Table 1 only the results of the optimal *F* acceptance factor are shown in each case when manual segmentation is used.

Table 1: Error rates table

k-means initialisation	HMM initialisation	Forward–backward	Viterbi
splitting	manual segmentation	7.70%	7.81%
	linear segmentation	5.62%	5.52%
k-means type	manual segmentation	8.75%	8.33%
	linear segmentation	7.91%	8.12%

From Table 1, it can be seen that the best result corresponds to the splitting initialisation for *k*-means and linear segmentation for HMM initialisation. Despite there being no significant differences between the splitting and *k*-means type convergence, the former clearly works better in recognition. No meaningful differences between the performance of the Viterbi and the forward–backward algorithms can be found. In general, splitting and linear segmentation works better than *k*-means type initialisation and manual segmentation.

7.4 Duration and threshold-based rejection performance

The next experiments were developed on a system of the same type as the best found in the previous section, i.e. using linear segmentation and splitting as initialisation algorithms for the HMM training and the codebook training. DB1 and DB2 were used for training and testing, respectively.

Four experiments were developed with four types of score. The inclusion of duration information with a threshold-based rejection is performed in two steps: first, only state duration is included (experiments 1 and 2), and second, state and word durations are included (experiments 3 and 4). These experiments are:

(*a*) *Experiment 1:* The log-score used for the utterance O in model λ is

$$x = \frac{\log (p(O \mid \lambda)) + \alpha_{sd} P_{sd}}{T} \tag{7}$$

In this case, the mean log-score per symbol includes the state duration log-score. State duration is included by the experimental weight α_{sd}. The optimal error rates for the different $p_i(d_i)$ distributions are: SD1 = 4.58% ($\alpha_{sd} = 0.7$), SD2 = 4.68% ($\alpha_{sd} = 0.7$), and SD3 = 5.20% ($\alpha_{sd} = 1.7$).

(*b*) *Experiment 2:* The duration information is simply added to the mean symbol score

$$x = \frac{\log (p(O \mid \lambda))}{T} + \alpha_{sd} P_{sd} \tag{8}$$

The optimal error rates for the different $p_i(d_i)$ distributions are: SD1 = 4.79% ($\alpha_{sd} = 0.03$), SD2 = 4.79% ($\alpha_{sd} = 0.03$), and SD3 = 5.20% ($\alpha_{sd} = 0.03$).

(*c*) *Experiment 3:* The same as experiment 1, but including word duration information

$$x = \frac{\log (p(O \mid \lambda)) + \alpha_{sd} P_{sd} + \alpha_{wd} P_{wd}}{T} \tag{9}$$

Word duration information is included as state duration in eqn. 1, using an experimental weight α_{wd}. An experiment (using SD1) was developed, obtaining that the error rate is an increasing function of α_{wd}.

(*d*) *Experiment 4:* The same as experiment 2, but including word duration

$$x = \frac{\log (p(O \mid \lambda))}{T} + \alpha_{sd} P_{sd} + \alpha_{wd} P_{wd} \tag{10}$$

The optimal error rate is 4.58% for $\alpha_{wd} = 0.05$ (using SD1).

These results show that it is better to include the state duration as in experiment 1 than as in experiment 2. The word duration is slightly useful in experiment 4 but not in experiment 3. In general, it does not imply any significant improvement. There are no important differences between SD1 and SD2, but SD3 yields the worst results in all the cases. This can be easily understood since it is not a Gaussian process (Fig. 5).

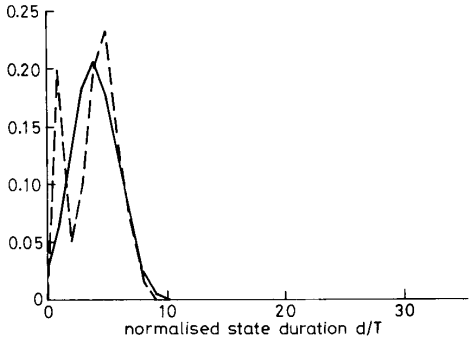

Fig. 5 *Histogram and Gaussian model of state duration*
State = 0, word = 'muñeca'.
- - - - histogram
——— Gaussian model

The rejection results of experiments 1 and 2 are depicted in Fig. 6, along with a rejection curve using a non-normalised log-score

$$x = \log (p(O \mid \lambda)) + \alpha_{sd} P_{sd} \quad (11)$$

The best rejection is obtained when the duration information is included in the mean symbol log-score (eqn. 7), and the threshold-based rejection works better for low rejections (where the curve slope is higher). The necessity of the temporal normalisation for the threshold-based rejection is also observed from the figure.

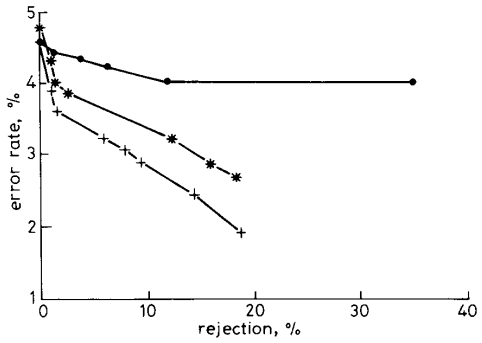

Fig. 6 *Error rate against rejection for log-scores*
SD1
+ experiment 1
* experiment 2
● non-normalised log score

8 Summary

A recognition system based on both vector quantisation and hidden Markov models was proposed. A vocabulary (the digits and six command words) specially thought for robot control applications was used. The objective was to obtain some improvements in this system performance, rather than to obtain a high performance, which is easily available using more centroids in the codebook, more models per word, etc.

Two initialisation methods for the k-means algorithm were presented: splitting and k-means type initialisations. Despite no important differences being found in the convergences of both methods, splitting clearly works better in recognition.

Two initialisation methods for the HMM training were also introduced, based on manual and linear segmentations. Using manual segmentation, the states have a physical meaning, since they roughly correspond to phonemes or other sound units, so the number of states is optimised in that sense. But it does not perform as well as linear segmentation. It seems clear that the amount of data that can be managed in a manual segmentation process can be enough for a speaker dependent system [12], but not for a speaker independent recognition system initialisation, for which it is preferable to use other initialisation procedures that involve all the training data.

Several HMM log-scores, including temporal normalisation and duration information, for utterance evaluation were tested. The best result was obtained using only state duration, including it in the mean log-score per symbol (eqn. 7). No significant differences were found between using normalised or non-normalised duration.

A threshold-based rejector (using the proposed log-scores) was used to diminish the error rate. It was shown that the temporal normalisation of score is basic for this rejection.

9 References

1 CLASS, F., KALTENMEIER, A., and KATTERFELDT, H.: 'Comparison of two continuous connected word recognition principles: hidden Markov modelling and dynamic time warping'. Proc. EUSIPCO-86, September 1986, **1,** pp. 553–556
2 LEVINSON, S.E., RABINER, L.R., and SONDHI, M.M.: 'An introduction to the application of the theory of probabilistic functions of a Markov process to automatic speech recognition', *Bell Syst. Tech. J.*, 1983, **62,** (4), pp. 1035–1074
3 RABINER, L.R., and JUANG, B.H.: 'An introduction to hidden Markov models', *IEEE ASSP Mag.*, January 1986, pp. 4–16
4 RABINER, L.R.: 'A tutorial on hidden Markov models and selected applications in speech recognition', *Proc. IEEE*, 1989, **77,** (2), pp. 257–286
5 MAKHOUL, J.: 'Linear prediction: a tutorial review' (IEEE PRESS, 1979), pp. 124–145
6 RABINER, L.R., and SCHAFER, R.W.: 'Digital processing of speech signals' (Prentice-Hall, Englewood Cliffs, 1978)
7 MARKEL, J.D., and GRAY, A.H.: 'Linear prediction of speech' (Springer-Verlag, Berlin, 1976)
8 GRAY, R.M., BUZO, A., GRAY, A.H., and MATSUYAMA, Y.: 'Distortion measured for speech processing', *IEEE Trans.*, 1980, **ASSP-28,** (4), pp. 367–376
9 GRAY, R.M.: 'Vector quantisation', *IEEE ASSP Mag.*, April 1984, pp. 4–29
10 BUZO, A., GRAY, A.H., GRAY, R.M., and MARKEL, J.D.: 'Speech coding based upon vector quantization', *IEEE Trans.*, 1980, **ASSP-28,** (5), pp. 562–574
11 MAKHOUL, J., ROUCOS, S., and GISH, H.: 'Vector quantization in speech coding', *Proc. IEEE*, 1985, **73,** (11), pp. 1551–1588
12 PEINADO, A.M., RUBIO, A.J., and LLORIS, A.: 'Reconocimiento de voz mediante modelos discretos de Markov'. Monografias del Departmento de Electronica, 13, May 1989
13 PEINADO, A.M., RAMESH, P., and ROE, D.B.: 'On the use of energy information for speech recognition using HMM'. Proc. EUSIPCO-90, September 1990, **2,** pp. 1243–1246
14 TOU, J.T., and GONZALEZ, R.C.: 'Pattern recognition principles' (Addison-Wesley, 1974)
15 RABINER, L.R., LEVINSON, S., and SONDHI, M.: 'On the applications of VQ and HMM to speaker-independent isolated recognition', *Bell Syst. Tech. J.*, 1983, **62,** (4), pp. 1075–1105
16 BOITE, R., LEICH, H., and ZANELLATO, G.: 'Isolated word recognition by hidden Marokov models'. Proc. EUSIPCO-86, September 1986, **1,** pp. 541–544
17 RABINER, L.R., and SAMBUR, M.R.: 'An algorithm for detecting the endpoints of isolated utterances', *Bell Syst. Tech. J.*, 1975, **54,** pp. 297–315