Multimodal Speech Processing Using Asynchronous Hidden Markov Models ⋆

Samy Bengio *

Dalle Molle Institute for Perceptual Artificial Intelligence (IDIAP), CP 592, rue du Simplon 4, 1920 Martigny, Switzerland

Abstract

This paper advocates that for some multimodal tasks involving more than one stream of data representing the same sequence of events, it might sometimes be a good idea to be able to *desynchronize* the streams in order to maximize their joint likelihood. We thus present a novel Hidden Markov Model architecture to model the joint probability of pairs of asynchronous sequences describing the same sequence of events. An Expectation-Maximization algorithm to train the model is presented, as well as a Viterbi decoding algorithm, which can be used to obtain the optimal state sequence as well as the alignment between the two sequences. The model was tested on two audio-visual speech processing tasks, namely speech recognition and text-dependent speaker verification, both using the M2VTS database. Robust performances under various noise conditions were obtained in both cases.

Key words: speech recognition, speaker verification, multimodal fusion, asynchronous fusion, joint EM estimation, HMM

1 Introduction

Hidden Markov Models (HMMs) are statistical tools that have been used successfully in the last 30 years to model difficult tasks such as speech recognition [1] or biological sequence analysis [2]. They are very well suited to handle

⋆ This research has been partially carried out in the framework of the European LAVA project, funded by the Swiss OFES project number 01.0412. It was also partially funded by the Swiss NCCR project (IM)2. The author would like to thank Stephane Dupont for providing the extracted visual features and the experimental protocol used in the paper.

* Corresponding author.
Email address: bengio@idiap.ch (Samy Bengio).

discrete or continuous sequences of varying sizes. Moreover, an efficient training algorithm (EM), which maximizes the likelihood of the data, is available, as well as an efficient decoding algorithm (Viterbi), which provides the optimal sequence of states (and the corresponding sequence of high level events) associated with a given sequence of low-level data.

On the other hand, multimodal information processing is currently a very challenging framework of applications including multimodal person authentication, multimodal speech recognition, multimodal event analyzers, etc. In that framework, the same sequence of events is represented not only by a single sequence of data but by a series of sequences of data, each of them coming eventually from a different *modality*: video streams with various viewpoints, audio stream(s), etc.

Two such tasks, which will be presented in this paper, involve multimodal speech processing using both a microphone and a camera recording a speaker simultaneously while he (she) speaks. It is well known that seeing the speaker's face in addition to hearing his (her) voice can often improve speech intelligibility, particularly in noisy environments [3], mainly thanks to the complementarity of the visual and acoustic signals. Previous solutions proposed for this kind of task can be subdivided into two categories [4]: *early integration*, where both signals are first modified to reach the same frame rate and are then modeled jointly, or *late integration*, where the signals are modeled separately and are combined later, during recognition or decision. While in the former solution, the alignment between the two sequences is decided *a priori*, in the latter, there is no explicit learning of the joint probability of the two sequences. An example of late integration is presented in [5], where the authors present a multi-stream approach for speech recognition where each stream is modeled by a different HMM, while decoding is done on a combined HMM (with various combination approaches proposed).

In [6], we presented the *Asynchronous Hidden Markov Model* (AHMM) that could learn the joint probability of pairs of sequences of data representing the same sequence of events, even when the events were not synchronized between the sequences. In fact, the model enables to *desynchronize* the streams by temporarily stretching one of them in order to obtain a better match between the corresponding frames. The model was applied in [6] to the problem of audio-visual speech recognition where sometimes lips start to move before any sound is heard for instance.

In this paper, we go into more detailed presentation and analysis of the model, and present results on two applications: audio-visual speech recognition, as in [6], and audio-visual text-dependent speaker verification. While in speech recognition the task is to transcribe audio-visual recordings into the corresponding sentence (sequence of words), in text-dependent speaker verification, the task is to let a genuine client enter a secured environment based on an audio-visual recording of a known text, while rejecting impostors trying to access the same system.

The paper is organized as follows: in the next section, the AHMM model is presented, followed by the corresponding training and decoding algorithms. Related models are then presented and implementation issues are discussed. Finally, experiments first on the audio-visual speech recognition task, and second on the audio-visual speaker verification task, both based on the M2VTS database, are presented. Finally, a short conclusion follows.

2 The Asynchronous Hidden Markov Model

Let us consider the case where one is interested in modeling the joint probability of two asynchronous sequences, denoted $X = x_1^T$ and $Y = y_1^S$ where T and S are respectively the length of sequences X and Y, with $S \leq T$ without loss of generality [1].

We are thus interested in modeling $p(x_1^T, y_1^S)$. Following the ideas introduced for Hidden Markov Models [1], we represent this distribution using a set of hidden variables [2] in order to decompose it into several simple factors. We thus first introduce a hidden variable Q which represents the *state* of the generating system as in the classical HMM formulation, and which is synchronized with the longest sequence. The state is a discrete variable. Let N be the number of different values this variable can take.

Moreover, since we know that S is smaller than T, let the system always emit x_t at time t but only sometimes emit y_s at time t, with $s \leq t$. Let us define $\epsilon(i,t) = P(\tau_t{=}s|\tau_{t-1}{=}s-1, q_t{=}i, x_1^t, y_1^s)$ as the probability that the system emits on sequence Y at time t while in state i. The additional hidden variable $\tau_t = s$ can be regarded as the alignment between Y and Q (and X which is always aligned with Q, by definition). Hence, an AHMM models $p(x_1^T, y_1^S, q_1^T, \tau_1^T)$.

Using these hidden variables, and assuming several independence hypotheses (see appendix A) we can factor the joint likelihood of the data and the hidden variable into several simple conditional distributions:

[1] In fact, we assume that for all pairs of sequences (X, Y), sequence X is always at least as long as sequence Y. If this is not the case, a straightforward extension of the proposed model is then necessary.

[2] These hidden variables represent some information which we assume exist in the model generating the observed data, but which is not available, *observed*.

- $P(q_t{=}i|q_{t-1}{=}j)$, the probability to go from state j to state i at time t,
- $p(x_t, y_s|q_t{=}i)$, the joint emission distribution of x_t and y_s, while in state i,
- $p(x_t|q_t{=}i)$, the emission distribution of x_t only, while in state i,
- $\epsilon(i,t)$, the probability to emit on both sequences while in state i at time t.

2.1 Likelihood Computation

One of the most important results obtained with HMMs was a simple yet efficient recursive procedure that could be used to compute the likelihood of the data only. We here propose a similar procedure for AHMMs. Using some independence assumptions (described in appendix A), a simple **forward procedure** can indeed be used to compute the joint likelihood of the two sequences, by introducing the following α intermediate variable which can be estimated recursively for each state and each possible alignment between the sequences X and Y:

$$\begin{aligned}\alpha(i,s,t) &= p(q_t{=}i, \tau_t{=}s, x_1^t, y_1^s) \qquad (1)\\ \alpha(i,s,t) &= \epsilon(i,t)p(x_t, y_s|q_t{=}i)\sum_{j=1}^{N} P(q_t{=}i|q_{t-1}{=}j)\alpha(j,s-1,t-1)\\ &\quad + (1-\epsilon(i,t))p(x_t|q_t{=}i)\sum_{j=1}^{N} P(q_t{=}i|q_{t-1}{=}j)\alpha(j,s,t-1)\ .\end{aligned}$$

It can then be used to compute the joint likelihood of the two sequences as follows:

$$\begin{aligned}p(x_1^T, y_1^S) &= \sum_{i=1}^{N} p(q_T{=}i, \tau_T{=}S, x_1^T, y_1^S) \qquad (2)\\ &= \sum_{i=1}^{N} \alpha(i,S,T)\ .\end{aligned}$$

2.2 Viterbi Decoding

In some applications, the likelihood of the data is not the real goal. In speech recognition for instance, one is interested in estimating the most likely path through the hidden variables that could have explained the data, in order to deduce the corresponding sequence of words associated with the states. Such an algorithm exists for normal HMMs and is called the Viterbi decoding algorithm [7].

With the same technique used to compute the likelihood, but replacing all the sums by max operators, we can derive a decoding algorithm for AHMMs similar to the classical Viterbi algorithm, which can then be used to obtain the most probable path along the sequence of states and alignments between X and Y. Again, the algorithm is based on a recursive equation as follows:

$$\begin{aligned} V(i,s,t) &= \max_{\tau_1^{t-1}, q_1^{t-1}} p(q_t{=}i, \tau_t{=}s, x_1^t, y_1^s) \\ &= \max \begin{cases} (\epsilon(i,t) p(x_t, y_s | q_t{=}i) \max\limits_j P(q_t{=}i | q_{t-1}{=}j) V(j, s-1, t-1), \\ (1-\epsilon(i,t)) p(x_t | q_t{=}i) \max\limits_j P(q_t{=}i | q_{t-1}{=}j) V(j, s, t-1)) \end{cases} . \end{aligned} \tag{3}$$

In order to obtain the best path, we simply need to compute every $V(i,s,t)$ recursively and keep for each of them the previous best predecessor state j and the corresponding alignment information. The best path is then obtained by backtracking from the best $V(i,S,T)$.

2.3 An EM Training Algorithm

An Expectation-Maximization (EM) [8] training algorithm can also be derived following the ideas developed for classical HMMs. We here sketch the resulting algorithm, without going into details.

EM is an iterative procedure for maximum likelihood estimation. Each iteration is composed of two steps: an *estimation* step and a *maximization* step. Both steps are based on the definition of an auxiliary function $A(\Theta;\hat{\Theta})$ which is the expectation, over the hidden variables, of the joint log likelihood of the observed (X and Y) and hidden (Q and τ) variables, with the expectation conditioned on the observed variables and the current value of the parameter set Θ:

$$A(\Theta;\hat{\Theta}) = E_{q,\tau}\left[\log p(x_1^T, y_1^S, q_1^T, \tau_1^T; \Theta) | x_1^T, y_1^S, \hat{\Theta}\right] . \tag{4}$$

It can be shown [8] that when the auxiliary function is maximized, the likelihood of the data is also maximized. In order to do so, we first estimate the expectation in the auxiliary function, by factorizing it (this is the E-step), and we then select the parameter set $\hat{\Theta}$ that maximizes the auxiliary function (this is the M-step). These steps are described in the following:

Backward Step: Similarly to the forward step based on the α variable used to compute the joint likelihood, a backward variable, β can also be defined and derived recursively as follows:

$$\beta(i,s,t) = p(x_{t+1}^T, y_{s+1}^S | q_t{=}i, \tau_t{=}s) \tag{5}$$

$$\beta(i,s,t) = \sum_{j=1}^{N} \epsilon(j,t+1) p(x_{t+1}, y_{s+1} | q_{t+1}{=}j) P(q_{t+1}{=}j | q_t{=}i) \beta(j,s+1,t+1)$$
$$+ \sum_{j=1}^{N} (1-\epsilon(j,t+1)) p(x_{t+1} | q_{t+1}{=}j) P(q_{t+1}{=}j | q_t{=}i) \beta(j,s,t+1)\ .$$

E-Step: The expectation in the auxiliary function can be factored (see appendix A) in various simple expectation terms. The expectations are based on the posterior probabilities of the hidden variables of the system, which can be computed using the forward and backward variables already defined.

Let $\alpha^1(i,s,t)$ be the part of $\alpha(i,s,t)$ when state i emits on Y at time t:

$$\alpha^1(i,s,t) = \epsilon(i,t) p(x_t, y_s | q_t{=}i) \sum_{j=1}^{N} P(q_t{=}i | q_{t-1}{=}j) \alpha(j,s-1,t-1) \tag{6}$$

and similarly, let $\alpha^0(i,s,t)$ be the part of $\alpha(i,s,t)$ when state i does not emit on Y at time t:

$$\alpha^0(i,s,t) = (1-\epsilon(i,t)) p(x_t | q_t{=}i) \sum_{j=1}^{N} P(q_t{=}i | q_{t-1}{=}j) \alpha(j,s,t-1)\ . \tag{7}$$

Then the posterior on state i when it emits on sequences X and Y is

$$P(q_t{=}i, \tau_t{=}s | \tau_{t-1}{=}s-1, x_1^T, y_1^S) = \frac{\alpha^1(i,s,t)\beta(i,s,t)}{P(x_1^T, y_1^S)}\ , \tag{8}$$

the posterior on state i when it emits on sequence X only is

$$P(q_t{=}i, \tau_t{=}s | \tau_{t-1}{=}s, x_1^T, y_1^S) = \frac{\alpha^0(i,s,t)\beta(i,s,t)}{P(x_1^T, y_1^S)}\ , \tag{9}$$

and the posterior on the transition between states i and j is

$$P(q_t{=}i, q_{t-1}{=}j | x_1^T, y_1^S) = \frac{P(q_t{=}i | q_{t-1}{=}j)}{P(x_1^T, y_1^S)} \cdot \tag{10}$$
$$\left(\begin{array}{l} \sum_{s=1}^{S} \alpha(j,s-1,t-1) p(x_t, y_s | q_t{=}i) \epsilon(i,t) \beta(i,s,t) + \\ \sum_{s=0}^{S} \alpha(j,s,t-1) p(x_t | q_t{=}i) (1-\epsilon(i,t)) \beta(i,s,t) \end{array} \right)\ .$$

As it can be seen, all these posteriors can be estimated efficiently using previously defined variables α and β.

M-Step: During the maximization step, the goal is to select a new set of parameters $\hat{\Theta}$ such that the auxiliary function is maximized. We are thus searching for the point where

$$\frac{\partial A(\Theta, \hat{\Theta})}{\partial \Theta} = 0 . \tag{11}$$

The M-step for Asynchronous HMMs is performed exactly as in classical HMMs: when the distributions are modeled by exponential functions such as Gaussian Mixture Models, then an exact maximization can be performed using the posteriors (equation 11 can be solved analytically). Otherwise, a Generalized EM is performed by gradient ascent, back-propagating the posteriors through the parameters of the distributions.

3 Related Models

The present AHMM model is related to the *Pair HMM* model [2], which was proposed to search for the best alignment between two DNA sequences. Pair HMMs were designed and used mainly for discrete sequences. Moreover, the architecture of Pair HMMs is such that a given state is designed to always emit either on one of the sequence OR on both sequences, while in the proposed AHMM model, each state can always emit either on one or on two sequences, depending on $\epsilon(i,t)$, which is learned. In fact, when $\epsilon(i,t)$ is deterministic and solely depends on i, we can indeed recover the Pair HMM model by slightly transforming the architecture.

It is also very similar to the asynchronous version of *Input/Output HMMs* [9], which was proposed for speech recognition applications. The main difference here is that in AHMMs both sequences are considered as output, while in Asynchronous IOHMMs one of the sequence (the shorter one, the output) is conditioned on the other one (the input). The resulting Viterbi decoding algorithm is thus different since in Asynchronous IOHMMs one of the sequences, the input, is known during decoding, which is not the case for AHMMs.

4 Implementation Issues

4.1 Time and Space Complexity

The proposed algorithms (either training or decoding) have a complexity of $O(N^2ST)$ where N is the number of states (and assuming the worst case with ergodic connectivity), S is the length of sequence Y and T is the length of sequence X. This can become quickly intractable if both X and Y are longer than, say, 1000 frames. Moreover, in the general case of modeling simultaneously and asynchronously M sequences of size S_i (instead of only two as presented in this paper), the complexity then becomes $O(N^2 \prod_{i=1}^{M} S_i)$.

It can fortunately be shortened when *a priori* knowledge is known about possible alignments between the streams (X and Y in the 2-stream case). For instance, one can force the alignment between x_t and y_s to be such that $|t - \frac{T}{S}s| < k$ where k is a constant representing the maximum stretching allowed between X and Y, which should not depend on S nor T. In that case, the complexity (both in time and space) becomes $O(N^2 T k)$, which is k times the usual HMM training/decoding complexity.

4.2 Distributions to Model

In order to implement this system, we thus need to model the following distributions:

- $P(q_t{=}i|q_{t-1}{=}j)$: the transition distribution. As in normal HMMs, this could be modeled using simple tables.
- $p(x_t|q_t{=}i)$: the emission distribution in the case where only X is emitted. As in normal HMMs, this could be modeled using Gaussian Mixture Models in the continuous case or simple tables in the discrete case.
- $p(x_t, y_s|q_t{=}i)$: the emission distribution in the case where both sequences are emitted. This distribution could be implemented in various forms, depending on the assumptions made on the data:
  · x and y are independent given state i:

$$p(x_t, y_s|q_t{=}i) = p(x_t|q_t{=}i)p(y_s|q_t{=}i) \tag{12}$$

  · y is conditioned on x:

$$p(x_t, y_s|q_t{=}i) = p(y_s|x_t, q_t{=}i)p(x_t|q_t{=}i) \tag{13}$$

  · the joint probability is modeled directly, eventually forcing some common parameters from $p(x_t|q_t{=}i)$ and $p(x_t, y_s|q_t{=}i)$ to be shared.

  In the experiments described in this paper, we have chosen the latter implementation, with no sharing except during initialization.
- $\epsilon(i, t) = P(\tau_t{=}s|\tau_{t-1}{=}s-1, q_t{=}i, x_1^t, y_1^s)$: the probability to emit on sequence y at time t on state i. With various assumptions, this probability could be represented as either independent on i, independent on s, independent on x_t and y_s. In the experiments described in this paper, we have chosen the latter implementation.

5 Experiments

In this section, we present two sets of experiments, one on speech recognition, and the second on text-dependent speaker verification. Both are based on the audio-visual M2VTS database [10], which contains 185 recordings of 37 subjects, each comprising acoustic and video signals of the subject pronouncing the French digits from zero to nine. The video consisted of 286x360 pixel color images with a 25 Hz frame rate, while the audio was recorded at 48 kHz using a 16 bit PCM coding. Although the M2VTS database is one of the largest databases of its type, it is still relatively small compared to reference audio databases used in speech recognition or speaker verification. Hence, in order to increase the significance level of the experimental results, a K-fold cross-validation method was used in both sets of experiments.

The audio data was down-sampled to 8khz and every 10ms a vector of 16 MFCC coefficients and their first derivative, as well as the derivative of the log energy was computed, for a total of 33 features. Each image of the video stream was coded using 12 shape features and 12 intensity features, as described in [5]. The method decomposes the lip shape and the gray-level intensities in the mouth region into a weighted sum of basis shapes (inner and outer lip contour) and basis intensities, respectively, using the Karhunen-Loeve expansion. These features, obtained by lip tracking, were normalized with respect to the mouth center, orientation, and width. The first derivative of each feature was also computed, yielding a total of 48 features.

5.1 Speech Recognition Experiments

The first set of experiments was done on a speech recognition task. Note that all the subjects always pronounced the same sequence of words but this information was not used during recognition [3].

The HMM topology was as follows: we used left-to-right HMMs for each instance of the vocabulary, which consisted of the following 11 words: zero, un, deux trois, quatre, cinq, six, sept, huit, neuf, silence. Each model had between 3 to 9 states including non-emitting *begin* and *end* states.

In each emitting state, there was 3 distributions: $P(x_t|q_t)$, the emission distribution of audio-only data, which consisted of a Gaussian mixture of 10 Gaussians (of dimension 33), $P(x_t, y_s|q_t)$, the joint emission distribution of audio and video data, which consisted also of a Gaussian mixture of 10 Gaus-

[3] Nevertheless, it can be argued that transitions between words could have been learned using the training data.

sians (of dimension 33+48=81), and $\epsilon(i,t)$, the probability that the system should emit on the video sequence, which was implemented as a table.

Training was done using the EM algorithm described in the paper. However, in order to keep the computational time tractable, a constraint was imposed on the alignment between the audio and video streams: we did not consider alignments where audio and video information were farther than 0.68 second from each other.

Comparisons were made between the AHMM (taking into account audio and video), and a normal HMM taking into account either the audio or the video only. We also compared the model with a normal HMM trained on both audio and video streams manually synchronized (each frame of the video stream was repeated in multiple copies in order to reach the same rate as the audio stream). Moreover, in order to show the interest of robust multimodal speech recognition, we injected various levels of noise in the audio stream during decoding (training was always done using clean audio). The noise was taken from the Noisex database [11]. It was injected in the data in order to reach segmental signal-to-noise (SNR) ratios of 10dB, 5dB and 0dB.

Note that all the hyper-parameters of these systems, such as the number of Gaussians in the mixtures, the number of EM iterations, or the minimum value of the variances of the Gaussians, were not tuned using the M2VTS dataset. They were taken from a previously trained model on a different task, Numbers'95 [12].

Figure 1 and Table 1 present the results in terms of *Word Error Rate*, a commonly used measure in the field of speech recognition, which takes into account three types of errors: the number of *insertions* (words that were decoded but did not really exist in the real transcript), *deletions* (words that were in the real transcript but were not decoded by the system) and *substitutions* (words that were decoded differently from the real transcript). As it can be seen, the AHMM yielded better results as soon as the noise level was significant. Moreover, it never deteriorated significantly (using a 95% confidence interval) under the level of the video stream, no matter the level of noise in the audio stream.

Comparing the *audio HMM* system with the *synchronized audio+video HMM* system, we can see that unless the noise level in the audio stream is very high, simply adding the video information in a naive linearly synchronized way did not enhance the performance, and in fact severely decreased it. It shows that adding information that is not properly synchronized may hurt the system's performance and should be avoided. It thus highlights the importance of the asynchronous HMM idea.

An interesting side effect of the model is to provide an optimal alignment be-

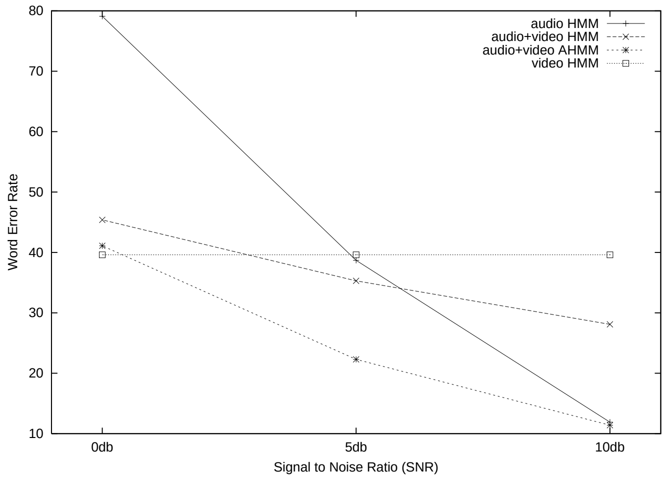

Fig. 1. Word Error Rates (in percent, the lower the better), of various systems under various noise conditions during decoding (from 10 to 0 dB additive noise). The proposed model is the AHMM using both audio and video streams.

Observations	Model	WER (%) and 95% CI		
		10 dB	5 dB	0 dB
audio	HMM	11.9 (± 4.7)	38.7 (± 7.1)	79.1 (± 5.9)
audio+video	HMM	28.1 (± 6.5)	35.3 (± 6.9)	45.4 (± 7.2)
audio+video	AHMM	11.4 (± 4.6)	22.3 (± 6.0)	41.1 (± 7.1)

Table 1
Word Error Rates (WER, in percent, the lower the better) and corresponding Confidence Intervals (CI, in parenthesis), of various systems under various noise conditions during decoding (from 10 to 0 dB additive noise). The proposed model is the AHMM using both audio and video streams. An HMM using the clean video data only obtains 39.6% WER (± 7.1).

tween the audio and the video streams. Figure 2 shows the alignment obtained while decoding sequence `cd01` on data corrupted with 10dB Noisex noise. It shows that the rate between video and audio is far from being constant (it would have followed the straight line followed by the HMM alignment) and hence computing the joint probability using the AHMM appears more informative than using a naive alignment and a normal HMM.

5.2 Text-Dependent Speaker Verification Experiments

The second set of experiments targeted a text-dependent speaker verification task. In that case, the goal was to accept genuine clients based on their audio-visual recording while rejecting impostors trying to access the system. We compared in these experiments 6 different models:

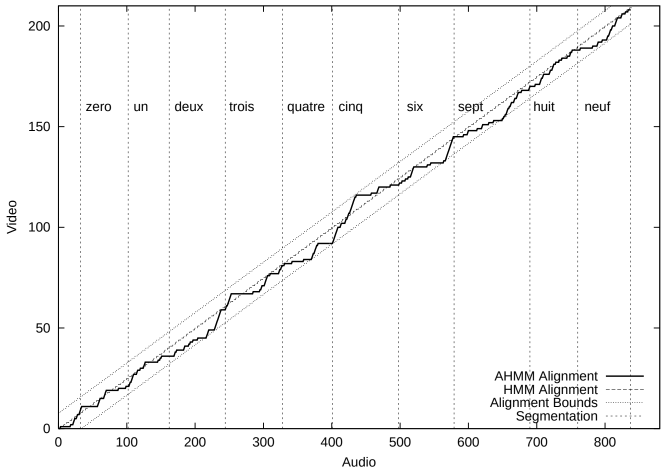

Fig. 2. Alignment obtained by the model between video and audio streams on sequence `cd01` corrupted with a 10dB Noisex noise. The vertical lines show the obtained segmentation between the words. The alignment bounds represent the maximum allowed stretch between the audio and the video streams.

- an AHMM trained on both audio and video data, as explained in the paper,
- an HMM trained on the fusion of audio and video data (by up-sampling correctly the video data to obtain the same number of frames in the two streams),
- an HMM trained on the audio data only,
- an HMM trained on the video data only,
- a Gaussian Mixture Model (GMM) trained on the audio data only,
- a fusion between the GMM on audio only and the HMM on video only. The fusion was performed using a multi-layer perceptron (see [13] for an introduction on MLPs) with the two scores as input.

In all the cases, we used the classical speaker verification technique, computing the difference between the log likelihood of the data given the client model and the log likelihood of the data given the world model (a model created with data not coming from the target client), and accepting the access when this difference was higher than a given threshold.

The K-fold cross-validation method used to assess the quality of the models was setup as follows: We used only 36 subjects, separated into 4 groups. For each subject, there was 5 different recording sessions. We used the first 2 sessions to create a client model, and the last 3 sessions to estimate the quality of the model. For each group, we used the other 3 groups to create a world model (using only the first 2 sessions per client). Moreover, for each client in one of the other three groups, we adapted a client specific model (using a simple MAP adaptation method [14]) from the world model (again using only the first 2 sessions of the client). Using these client-specific models, we selected a global threshold such that it yielded an Equal Error Rate (EER, when the False Acceptance Rate, FAR, is equal to the False Rejection Rate, FRR). Finally, we adapted (using MAP again) a client-specific model from the world model for each client of the current test group and computed the Half Total Error Rate (HTER, the average of the FAR and the FRR) on the last three accesses of each test client using the global threshold previously found. Hence, all results presented here can be seen as unbiased since no parameter (including the threshold) was computed using the test accesses.

The architecture of the models (number of states, number of Gaussians, etc) were the same as in the speech recognition experiments. The GMM models used a silence removal technique based on an unsupervised bi-Gaussian method in order to remove all non-informative frames [15].

As in the speech recognition experiments, we injected various levels of noise in the audio stream during test accesses (training was always done using clean audio). The noise was the same as in the speech recognition experiments, and SNR were also set to 10db, 5db, and 0db.

Figure 3 presents the results. For each method at each level of noise injected in the audio stream, we present the Half Total Error Rate (HTER), a measure often used to assess the quality of a verification system. As it can be seen, the AHMM yielded better and more stable results as soon as the noise level in the audio stream was significant. For almost clean data, the performance of the GMM using the audio stream only as well as the one of the fusion of the score of the GMM with the score of the video HMM model were better, but quickly deteriorated with the addition of noise.

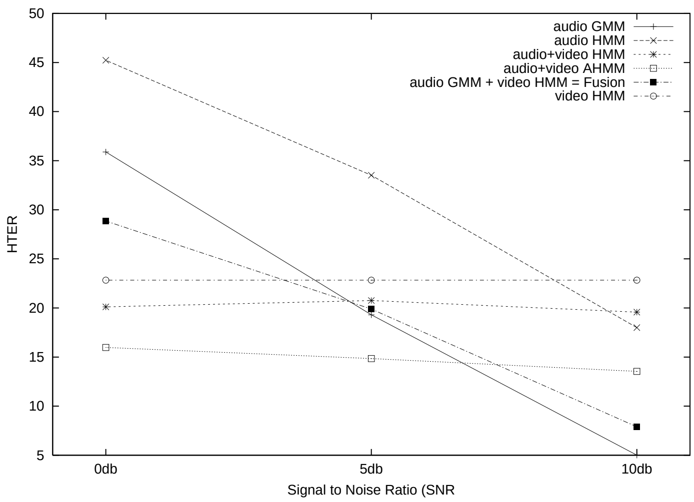

Fig. 3. HTER (the lower the better), of various systems under various noise conditions during test (from 10 to 0 dB additive noise). The proposed model is the AHMM using both audio and video streams.

6 Conclusion

In this paper, we have presented a novel asynchronous HMM architecture to handle multiple sequences of data representing the same sequence of events. The model was inspired by two other well-known models, namely Pair HMMs and Asynchronous IOHMMs. An EM training algorithm was derived as well as a Viterbi decoding algorithm. Continuous speech recognition and text-dependent speaker verification experiments were performed on a multimodal database, yielding significant improvements on noisy audio data. Various propositions were made to implement the model but only the simplest ones were tested in this paper. Other solutions should thus be investigated soon. Extension of the model to more than two streams is probably also very interesting but care should then be taken to keep the computational time tractable.

A Details about the EM Algorithm

A.1 Assumptions

The usual HMM assumptions are used here: the probability of the observation given the state does not depend on anything else than the state, and the probability of being in a state at time t depends only on the state the system was at time $t-1$:

$$p(x_t|q_t{=}i, x_1^{t-1}, y_1^s, \tau_{t-1}{=}s) \stackrel{\text{def}}{=} p(x_t|q_t{=}i) \tag{A.1}$$

$$p(x_t, y_s|q_t{=}i, x_1^{t-1}, y_1^{s-1}, \tau_{t-1}{=}s-1) \stackrel{\text{def}}{=} p(x_t, y_s|q_t{=}i) \tag{A.2}$$

$$P(q_t{=}i|q_{t-1}{=}j, x_1^{t-1}, y_1^{s-1}, \tau_{t-1}{=}s-1) \stackrel{\text{def}}{=} P(q_t{=}i|q_{t-1}{=}j) \tag{A.3}$$

A.2 Derivation of the Forward Step

The forward step describes how to compute $\alpha(i, s, t)$. The derivation is very similar to the classical HMM forward derivation, with the special case that handles the τ_t variable, which contains the alignment between x and y.

$$
\begin{aligned}
\alpha(i,s,t) &= P(q_t{=}i, \tau_t{=}s, x_1^t, y_1^s, u_{t-1}{=}s-1) + P(q_t{=}i, \tau_t{=}s, x_1^t, y_1^s, u_{t-1}{=}s) && \text{(A.4)}\\
\alpha(i,s,t) &= P(\tau_t{=}s|q_t{=}i, x_1^t, y_1^s, u_{t-1}{=}s-1)P(q_t{=}i, x_1^t, y_1^s, u_{t-1}{=}s-1)\\
&\quad + P(\tau_t{=}s|q_t{=}i, x_1^t, y_1^s, u_{t-1}{=}s)P(q_t{=}i, x_1^t, y_1^s, u_{t-1}{=}s) && \text{(A.5)}\\
\alpha(i,s,t) &= \epsilon(i,t)p(x_t, y_s|q_t{=}i, x_1^{t-1}, y_1^{s-1}, u_{t-1}{=}s-1)\\
&\qquad P(q_t{=}i, x_1^{t-1}, y_1^{s-1}, u_{t-1}{=}s-1)\\
&\quad + (1-\epsilon(i,t))p(x_t|q_t{=}i, x_1^{t-1}, y_1^s, u_{t-1}{=}s)\\
&\qquad P(q_t{=}i, x_1^{t-1}, y_1^s, u_{t-1}{=}s) && \text{(A.6)}\\
\alpha(i,s,t) &= \epsilon(i,t)p(x_t, y_s|q_t{=}i)\\
&\qquad P(q_t{=}i, x_1^{t-1}, y_1^{s-1}, u_{t-1}{=}s-1)\\
&\quad + (1-\epsilon(i,t))p(x_t|q_t{=}i)\\
&\qquad P(q_t{=}i, x_1^{t-1}, y_1^s, u_{t-1}{=}s) && \text{(A.7)}\\
\alpha(i,s,t) &= \epsilon(i,t)p(x_t, y_s|q_t{=}i)\\
&\qquad \sum_{j=1}^{N} P(q_t{=}i|q_{t-1}{=}j, x_1^{t-1}, y_1^{s-1}, u_{t-1}{=}s-1)P(q_{t-1}{=}j, x_1^{t-1}, y_1^{s-1}, u_{t-1}{=}s-1)\\
&\quad + (1-\epsilon(i,t))p(x_t|q_t{=}i)\\
&\qquad \sum_{j=1}^{N} P(q_t{=}i|q_{t-1}{=}j, x_1^{t-1}, y_1^{s-1}, u_{t-1}{=}s)P(q_{t-1}{=}j, x_1^{t-1}, y_1^s, u_{t-1}{=}s) && \text{(A.8)}\\
\alpha(i,s,t) &= \epsilon(i,t)p(x_t, y_s|q_t{=}i)\sum_{j=1}^{N} P(q_t{=}i|q_{t-1}{=}j)\alpha(j, s-1, t-1)\\
&\quad + (1-\epsilon(i,t))p(x_t|q_t{=}i)\sum_{j=1}^{N} P(q_t{=}i|q_{t-1}{=}j)\alpha(j, s, t-1) && \text{(A.9)}
\end{aligned}
$$

A.3 Auxiliary Function Leading to EM and Viterbi

As in the normal HMM derivation, an auxiliary function A is introduced which is defined as the expectation, over the hidden variables, of the joint log-likelihood of the observed variables and the hidden variables, when the expectation is conditioned on the observed variables and the current value of the parameters:

$$
A(\Theta; \hat{\Theta}) = E_{q,\tau}\left[\log p(x_1^T, y_1^S, q_1^T, \tau_1^T; \Theta)|x_1^T, y_1^S, \hat{\Theta}\right] \tag{A.10}
$$

$$A(\Theta;\hat{\Theta}) = \sum_{t=1}^{T}\sum_{i=1}^{N}\left(\begin{array}{l}\sum_{s=1}^{S} E[q_t{=}i, \tau_t{=}s|\tau_{t-1}{=}s-1, x_1^T, y_1^S]\log p(y_s, x_t, \tau_t{=}s|q_t{=}i, \tau_{t-1}{=}s-1)+\\ \sum_{s=1}^{S} E[q_t{=}i, \tau_t{=}s|\tau_{t-1}{=}s, x_1^T, y_1^S]\log p(x_t, \tau_t{=}s|q_t{=}i, \tau_{t-1}{=}s)+\\ \sum_{j=1}^{N} E[q_t{=}i, q_{t-1}{=}j|x_1^T, y_1^S]\log P(q_t{=}i|q_{t-1}{=}j)\end{array}\right)$$

A.4 Derivation of the Backward Step

As in the normal HMM derivation, on top of the forward variable $\alpha(i,s,t)$, we also need a backward variable $\beta(i,s,t)$ which basically describes the probability to emit the rest of the two sequences, when we are in a given state and a given alignment.

$$\beta(i,s,t) = p(y_{s+1}^S, x_{t+1}^T|q_t{=}i, \tau_t{=}s) \tag{A.11}$$

$$= \sum_{j=1}^{N} P(y_{s+1}^S, x_{t+1}^T, q_{t+1}{=}j|q_t{=}i, \tau_t{=}s) \tag{A.12}$$

$$= \sum_{j=1}^{N} P(y_{s+1}^S, x_{t+1}^T, q_{t+1}{=}j, \tau_{t+1}{=}s{+}1|q_t{=}i, \tau_t{=}s)$$
$$+ \sum_{j=1}^{N} P(y_{s+1}^S, x_{t+1}^T, q_{t+1}{=}j, \tau_{t+1}{=}s|q_t{=}i, \tau_t{=}s) \tag{A.13}$$

$$= \sum_{j=1}^{N}\left(\begin{array}{l} P(q_{t+1}{=}j|q_t{=}i)\cdot\\ p(y_{s+1}, x_{t+1}|\tau_{t+1} = s{+}1, \tau_t = s, q_{t+1}{=}j)\cdot\\ P(\tau_{t+1}{=}s{+}1|\tau_t{=}s, q_{t+1}{=}j)\cdot\\ p(y_{s+2}^S, x_{t+2}^T|q_{t+1}{=}j, \tau_{t+1}{=}s{+}1)\end{array}\right)$$
$$+ \sum_{j=1}^{N}\left(\begin{array}{l} P(q_{t+1}{=}j|q_t{=}i)\cdot\\ P(\tau_{t+1}{=}s|\tau_t{=}s, q_{t+1}{=}j)\cdot\\ p(x_{t+1}|, q_{t+1}{=}j)\cdot\\ p(y_{s+1}^S, x_{t+2}^T|q_{t+1}{=}j, \tau_{t+1}{=}s)\end{array}\right) \tag{A.14}$$

$$= \sum_{j=1}^{N} \epsilon(j,t+1)p(x_{t+1}, y_{s+1}|q_{t+1}{=}j)P(q_{t+1}{=}j|q_t{=}i)\beta(j,s+1,t+1)$$
$$+ \sum_{j=1}^{N}(1-\epsilon(j,t+1))p(x_{t+1}|q_{t+1}{=}j)P(q_{t+1}{=}j|q_t{=}i)\beta(j,s,t+1) \tag{A.15}$$

In order to maximize the likelihood, we in fact maximize the value of the auxiliary function. We thus use the same techniques as in the normal EM for HMMs: for each parameter, we search for the value such that the derivative of the auxiliary function with respect to this parameter is equal to 0. For instance, in order to select the new mean $\hat{\mu}_{i,j}$ of Gaussian i in state j in the joint emission model, we need to search for the value $\hat{\mu}_{i,j}$ such that

$$\frac{\partial A(\Theta, \hat{\Theta})}{\partial \mu_{i,j}} = 0 \tag{A.16}$$

References

[1] L. R. Rabiner, A tutorial on hidden markov models and selected applications in speech recognition, Proceedings of the IEEE 77 (2) (1989) 257–286.

[2] R. Durbin, S. Eddy, A. Krogh, G. Michison, Biological Sequence Analysis: Probabilistic Models of proteins and nucleic acids, Cambridge University Press, 1998.

[3] W. H. Sumby, I. Pollak, Visual contributions to speech intelligibility in noise, Journal of the Acoustical Society of America 26 (1954) 212–215.

[4] A. Summerfield, Lipreading and audio-visual speech perception, Philosophical Transactions of the Royal Society of London, Series B 335 (1992) 71–78.

[5] S. Dupont, J. Luettin, Audio-visual speech modelling for continuous speech recognition, IEEE Transactions on Multimedia 2 (2000) 141–151.

[6] S. Bengio, An asynchronous hidden markov model for audio-visual speech recognition, in: S. Becker, S. Thrun, K. Obermayer (Eds.), Advances in Neural Information Processing Systems, NIPS 15, 2003.

[7] A. Viterbi, Error bounds for convolutional codes and an asymptotically optimum decoding algorithm, IEEE Transactions on Information Theory (1967) 260–269.

[8] A. P. Dempster, N. M. Laird, D. B. Rubin, Maximum-likelihood from incomplete data via the EM algorithm, Journal of Royal Statistical Society B 39 (1977) 1–38.

[9] S. Bengio, Y. Bengio, An EM algorithm for asynchronous input/output hidden markov models, in: Proceedings of the International Conference on Neural Information Processing, ICONIP, Hong Kong, 1996.

[10] S. Pigeon, L. Vandendorpe, The M2VTS multimodal face database (release 1.00), in: Proceedings of the First International Conference on Audio- and Video-based Biometric Person Authentication ABVPA, 1997.

[11] A. Varga, H. Steeneken, M. Tomlinson, D. Jones, The noisex-92 study on the effect of additive noise on automatic speech recognition, Tech. rep., DRA Speech Research Unit (1992).

[12] R. A. Cole, M. Noel, T. Lander, T. Durham, New telephone speech corpora at CSLU, in: Proceedings of the European Conference on Speech Communication and Technology (EUROSPEECH), Vol. 1, 1995, pp. 821–824.

[13] C. Bishop, Neural Networks for Pattern Recognition, Clarendon Press, Oxford, 1995.

[14] D. A. Reynolds, T. F. Quatieri, R. B. Dunn, Speaker verification using adapted gaussian mixture models, Digital Signal Processing 10 (1–3).

[15] S. Bengio, J. Mariéthoz, S. Marcel, Evaluation of biometric technology on XM2VTS, Technical Report IDIAP-RR 01-21, IDIAP (2001).