

# SHORT-TIME VITERBI FOR ONLINE HMM DECODING : EVALUATION ON A REAL-TIME PHONE RECOGNITION TASK

Julien Bloit, Xavier Rodet

Ircam, CNRS-UMR9912  
1, place I.Stravinsky, 75004 Paris

## ABSTRACT

In this paper we present the implementation of an online HMM decoding process. The algorithm derives an online version of the Viterbi algorithm on successive variable length windows, iteratively storing portions of the optimal state path. We explicit the relation between the hidden layer's topology and the applicability and performance prediction of the algorithm. We evaluate the validity and performance of the algorithm on a phone recognition task on a database of continuous speech from a native French speaker. We specifically study the latency-accuracy performance of the algorithm.

**Index Terms**— Viterbi decoding, Real time systems, Hidden Markov models, Phone recognition.

## 1. INTRODUCTION

Artistic paradigms often involve signal-to-symbolic sub-tasks such as musical score to audio alignment, or gesture following [1]. In such cases, the system's latency is critical for subsequent actions to happen in an interactive way.

Finding the optimal state sequence with respect to the *Maximum a posteriori* (MAP) criterion is efficiently done with the Viterbi algorithm by tracing back through a matrix of back-pointers, starting from the end of the sequence. However, standard implementation of this method is unsuitable in the case of a streamed input for which there is potentially no ending to the sequence. One way to cope with this problem consists in applying the Viterbi algorithm on successive windows and outputting only the first states of the decoded path [2], [3]. An interesting aspect of this method is that the introduced delay can be set beforehand to a maximum value. However such solutions generally lead to suboptimal decoding.

An alternative approach consists in comparing partial path hypotheses on an expanding window until they converge towards the same solution. Such a method was implemented in a video target-tracking application [4] as well as on conditional random fields, where the look-ahead window was dynamically set so as to balance a latency/accuracy trade off measure [5]. However, no path convergence condition is clearly outlined in these works.

In this paper, we explicitly study the influence of the model's topology on potential paths convergence, accuracy and latency. We further combine both approaches by forcing a suboptimal state label output when the latency exceeds a predefined threshold. Quantitative evaluation is carried out on a real-time phone recognition task.

## 2. VITERBI DECODING

### 2.1. Standard Viterbi

Let us consider a given model and observation sequence pair  $\{\lambda, \mathbf{O}_T\}$ , where  $\lambda$  is parametrized by a state-space  $\mathcal{S}$  of size  $K$ , a prior state distribution  $\pi$ , a transition matrix  $A$  and a state emission density  $b_i(O_t) = P(s_t = i | O_t, \lambda)$ . The decoding step of a recognition system consists in retrieving a state sequence that maximizes the maximum a posteriori probability [6]. Such a sequence is referred to as the optimal path  $\mathbf{s}^*$ . The Viterbi algorithm is a method that yields the best path solution, with two main iterative steps :

1. a time-synchronous forward pass to update the partial likelihoods  $\delta_t(i), \forall i \in \mathcal{S}$  : the score of the best path ending in state  $i$  at time  $t$ . The best state predecessors are stored in a matrix of back-pointers  $\psi_t(i)$ .
2. a backtracking on  $\psi_t(i)$ , starting from the state with the highest score at time  $T$

The backtrack step implies that the algorithm is not time-synchronous since the last observation frame at time  $T$  is needed in order to decode the global optimal state path  $\mathbf{s}^*$  from  $T$  back to the first time index.

### 2.2. Short-time Viterbi

Let us define  $\mathbf{s}(a, b, i)$  as the state sequence obtained by computing  $\delta$  and  $\psi$  values on an observation window delimited by time indices  $a$  and  $b$  (such that  $a < b$ ) and backtracking from an arbitrary state  $i$  at time  $b$ . We further refer to such a state sequence as a *local path*. the set of local paths  $\{\mathbf{s}(a, b, i), \forall i \in \mathcal{S}\}$  are identical on an initial path portion, from  $t = a$  up to an instant  $\tau < T$ , which we refer to as a *fusion point* (as referred to in a Dynamic Time Warping context

[7]). A fusion point  $\tau$  has the attractive following property : the local paths up to  $\tau$  are always identical to the global Viterbi path between  $a$  and  $\tau$ .

*Proof:* Let us consider a set of local paths  $\{s(a, b, i), \forall i \in \mathcal{S}\}$  for which a fusion point arises at time  $\tau$  :

1. All back-pointers stay locally unchanged as the forward recursion in step-2 progresses : all  $\psi_b(i) \forall i$  computed at a given time  $b$  are not affected by further iterations of the algorithm at  $t > b$ .
2. as a consequence, all backtracks from  $t > b$  reaching state  $i$  at time  $b$  yield the same path portion between  $a = 1$  and  $\tau$ .
3. the local paths set represents backtracks from all reachable states at time  $b$ , including  $s_t^*$
4. it follows that  $s_t(a, b, \forall i) = s_t^*, \forall t = \{a, \dots, \tau\}$ .

From this observation, it is straightforward to derive an online short-time Viterbi (STV) algorithm, based on local Viterbi decodings between successive fusion points, used as left bounds for variable size observation windows :

---

#### Algorithm 1 Short-time Viterbi

---

- 1: Init. :  $a = 0, b = 0$
  - 2: **for** each new frame at time  $b$  **do**
  - 3:   compute  $s_t(a, b, i), \forall i \in \mathcal{I}(a, b, \lambda)$
  - 4:   **if** a fusion point  $\tau$  is found s.t.  $\tau > a$  **then**
  - 5:     output  $s_{a, \tau}^{\boxtimes}$
  - 6:      $a = \tau$
  - 7:   **end if**
  - 8: **end for**
- 

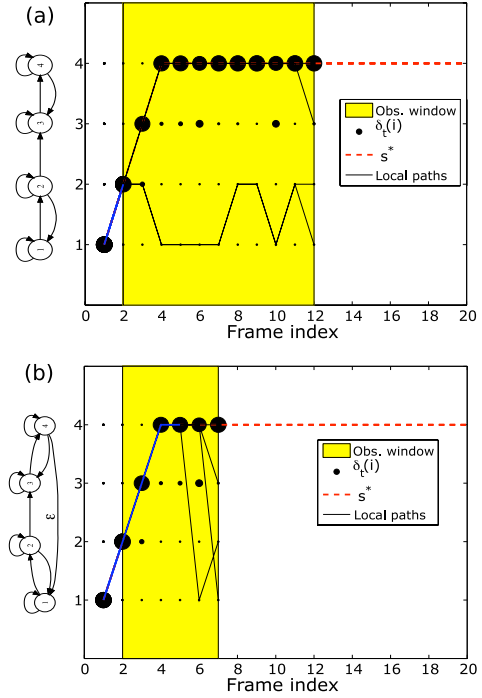
where  $s_{t, t'}^{\boxtimes}$  is the path found with STV for frames from  $t$  to  $t'$ , and  $\mathcal{I}(a, b, \lambda)$  denotes the subset of reachable states at time  $b$  for a given model  $\lambda$  and given an initial state at time  $a$ .

### 3. INFLUENCE OF THE MODEL'S TOPOLOGY

#### 3.1. Necessary Condition for Fusion Points

The above algorithm assumes that a fusion point occurs inside a time interval  $[a, b]$  smaller than  $[1, T]$ . However this is not always allowed by the topology of the model. Consider the execution of the STV algorithm on a toy model and sequence pair  $\{\lambda, \mathbf{O}_T\}$  represented on Figure 1.a). Despite the simplicity of the model and the strong  $\delta_t(i)$  scores on the optimal path state (depicted as larger dots), local paths cannot converge towards state 4 (on optimal path) because backtracking from states 1 and 2 can only lead to states 1 and 2. Thus the algorithm is doomed to stay stuck as the observation window expands until the final observation  $T$ .

If we now consider the almost same pair  $\{\lambda', \mathbf{O}_T\}$ , where the only modification made to the model consists in an additional, arbitrarily small transition probability  $\epsilon$  from state 4



**Fig. 1.** Toy example : on figure a), the set of reachable states after time  $b=12$  are not connex, thus the local paths cannot fuse. On figure b), the model topology was made connex, thus allowing a fusion point to arise at  $\tau = 5$

to state 1, as shown on Figure 1.b). When the observation window is large enough, all states in  $\mathcal{I}(a, b, \lambda')$  are now connex, i.e. any state can be reached from any other, allowing backtracks towards states with the highest  $\delta_t(i)$  scores, and eventually, the fusing of local paths as shown in Figure 1.b), where a fusion point arises at  $t = 5$ .

This example illustrates that a necessary condition for a fusion point to arise is that all states in  $\mathcal{I}(a, b, \lambda)$  have to be connex.

#### 3.2. Influence on Latency

Consider the state latency  $lat(s_t^{\boxtimes})$  as the time interval (in number of frames) needed for the STV algorithm to output the label of frame  $t$ . Using the previous notation,  $lat(s_t^{\boxtimes}) = b - t$ . The goal of a real-time algorithm is to minimize this value for each  $t$ . Once again, the model's topology influences our algorithm's behavior by setting a lower bound to the minimum possible latency. This derives directly from the connectivity constraint discussed in the previous paragraph.

Let  $L = b - a$  be the length of the observation window. The earliest state that can be decoded is at time  $a + 1$ , and consequently the minimum latency for this state is  $L - 1$ . A fusion point can occur at  $a + 1$  only if all states in  $\mathcal{I}(a, b, \lambda)$  are connex. Let the distance  $d(i, j)$  be the transition count

of the shortest backward path from state  $j$  to  $i$  according to matrix  $A$  and let  $D_{\mathcal{I}}$  be the set of these lengths for any pair  $(i, j)$  in  $\mathcal{I}$ . In the most adverse case,  $L$  will need to be at least  $\max(D_{\mathcal{I}}) + 1$  long to decode the state at time  $a + 1$ . This implies that in some cases, even with very likely data, the latency will have a minimum lower bound equal to  $\max(D_{\mathcal{I}})$ . For example, in the model of Figure 1.b), the two largest distances are  $d(1, 4)=3$  and  $d(3, 2)=3$ . So we can affirm that in the general case, we can not achieve optimal decoding with a better latency than 3 frames.

## 4. EXPERIMENTATIONS

### 4.1. Database and Models

We evaluated the performance of STV on a real-time phoneme recognition task. We used a corpus of continuous sentences in French, pronounced by a native speaker. The database consists in 3794 sentences split to 3640 sentences for the training set (131967 phonemes), and to 154 sentences (3134 phonemes) for the test set. The original sound files were sliced into 25 ms windows every 5 ms. For each resulting frame, we extracted 39 features consisting in 13 MFCC values along with their first and second order time derivatives.

A set of 37 monophone models (including a short-pause and a silence model) were learned on the training set by maximum likelihood estimation using the HTK tools [8]. Each monophone was modeled by a five states left-right HMM, except for the short-pause model that has only one state. The observation densities are modeled using Gaussian mixture models with diagonal covariance and  $J$  components per state, yielding a total of eight sets of models for  $J=1, 2, 4, \dots, 128$ . We also trained a bigram language model over the phone label files of the training set using HLSTATS tool from HTK. The reference model  $\lambda_{ref}$  was obtained by combining the language model with the phoneme models.

### 4.2. Topology Modifications

In order to make the model a valid candidate for the STV algorithm evaluation, we added several transitions to the topology of the original model  $\lambda_{ref}$ , to ensure connexity among the models states. As mentioned in section 3.2, there should be a relation between the connexity distance and the resulting latency. We thus derived two modified models from  $\lambda_{ref}$ : one that guarantees connexity by adding transitions at the phone-level (thus denoted  $\lambda_P$ , with  $\max(D_S) = 5$ ), and another ( $\lambda_S$ ) that adds transitions at the state-level ( $\max(D_S) = 1$ ).

The topology of  $\lambda_P$  is shown on Figure 2. Each state embeds a monophone model. For the sake of clarity, the bigram model's probabilities are not represented. The two inner non-emitting states represent the inter-monophone transitions. The backward transitions  $\epsilon$  are the topology modifications (we choose the value of  $\epsilon$  as the double-precision value in Matlab)

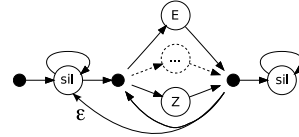


Fig. 2. Phone-level topology of model  $\lambda_P$ , including additional  $\epsilon$  backward transitions.

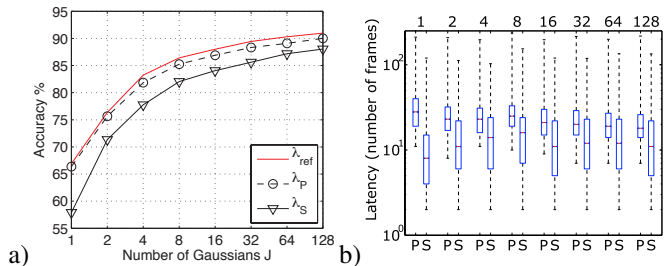


Fig. 3. a) Phone recognition scores : comparison of the unmodified models  $\lambda_{ref}$  behavior with two modified model sets  $\lambda_P$  and  $\lambda_S$ . b) Latencies for different model structures pairs, with increasing number of Gaussians. P and S letters refer to  $\lambda_P$  and  $\lambda_S$  models sets respectively.

### 4.3. Recognition Performance

Phone recognition results are evaluated using the reference transcriptions, without taking timing information into account. After alignment of the resulting phone transcription with the reference, the accuracy score [8] is computed :

$$\bullet \text{ Accuracy (\%)} : \frac{N-D-S-I}{N} . 100$$

where  $N, D, S$  and  $I$  are the number of phonemes, the number of deletion errors, of substitution errors and of insertion errors, respectively. Using this score measure, the recognition performance of our algorithm can be compared to the offline case. The resulting curves in Figure 3.a) show that online recognition with  $\lambda_P$  performs almost as well as in the offline case. As one could expect,  $\lambda_S$  models lead to more insertion errors.

### 4.4. Latency Performance

For each utterance decoding, we stored each fusion point information as a triplet of timestamps values  $(a, \tau, b)$ , thus allowing to compute the following decoding latency measure :

$$lat(s_t^{\otimes}) = b - t, \forall t \in [a, \tau] \quad (1)$$

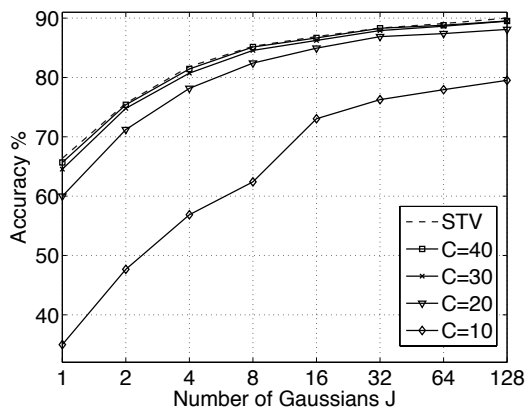
Figure 3.b) shows boxplots of latency values for model sets of topologies  $\lambda_P$  and  $\lambda_S$ . For models  $\lambda_P$ , as the number of Gaussians per state increases, the latency tends towards a decrease after the bump at  $J = 8$ . The utterances lengths in the test set range from 249 to 1470 frames, with a mean equal to 524 frames. The maximum found latency across all results

is 236. This means that all online decodings managed to find a fusion point. Looking at the minimum latencies obtained with  $\lambda_P$ , a floor value equal to 7 is reached for  $J = 32$  up to  $J = 128$ .

For models  $\lambda_S$ , although the number of Gaussians seems to be uncorrelated to any latency tendency, the values are comparatively concentrated below the latency values of  $\lambda_P$  models, in a significant way. Furthermore, a floor latency value of 2 is reached for all  $J$  model sets. The two floor latencies are consistent with their theoretical minimum values (cf. section 4.2).

#### 4.5. Hard Constraint on Latency

In practical interactive settings, a critical maximum latency  $C$  can be imposed on the system. We modified the STV algorithm in order to force an output for time  $a$  when  $(b-a) > C$  and no fusion point has yet been detected. The state at time  $a$  is then chosen on the local path that maximizes  $\delta_b(i), \forall i \in \mathcal{I}$ .



**Fig. 4.** Accuracy curves for four imposed maximum latencies, and an increasing number of Gaussians per state. The accuracy of STV alone is reproduced for comparison.

In Figure 4, we plotted accuracy scores obtained with the combination of STV and a forced output on a  $\lambda_P$  connex model for different values of  $C$ . The curves clearly show a latency/accuracy trade-off : as  $C$  gets shorter, accuracy decreases because of the suboptimal decoding procedure employed when forcing the output.

#### 5. CONCLUSION AND PERSPECTIVE

This experimental study provides supporting evidence for the applicability of the STV decoding algorithm, under identified topological constraints on the model. The quantitative results we gathered showed that it is possible to operate modifications on a model and make it obey these constraints. We proposed a way of satisfying these, by ensuring the model's states connexity, at two different model level. It appears that

as the connexity lengths reduce, the lower bound on best possible latency decreases. This happens at the expense of a joint decrease in accuracy, caused by more insertion errors. Finally, we have evaluated a proposed algorithm combining optimal online decoding with a hard-latency constraint, showing promising performances.

We have implemented this work in the Max/MSP realtime environment for real-life testing in musical interactive setups. Future work will focus on extensive testing with other models, in the context of singing voice or gesture recognition.

#### 6. ACKNOWLEDGMENTS

We wish to acknowledge Pierre Lanchantin for the phone models, and Nicolas Rasamimanana for valuable advices.

#### 7. REFERENCES

- [1] N. Orio, S. Lemouton, and D. Schwarz, "Score following : State of the art and new developments," in *NIME*, 2003.
- [2] A. Seward, "Low-latency incremental speech transcription in the synface project," in *EUROSPEECH*, Geneva, 2003.
- [3] M. Ryynaenen and A. Klapuri, "Automatic bass line transcription from streaming polyphonic audio," in *ICASSP*, 2007.
- [4] H. Ardo, K. Astrom, and R. Berthilsson, "Real time viterbi optimization of hidden markov models for multi target tracking," in *IEEE Workshop on Motion and Video Computing*, Feb 2007.
- [5] M. Narasimhan, P. Viola, and M. Shilman, "Online decoding of markov models under latency constraints," in *International Conference on Machine Learning*, 2006.
- [6] L. Rabiner, "A tutorial on hidden markov models and selected applications in speech recognition," in *Proceedings of the IEEE*, 1989.
- [7] H. Kaprykowsky and X. Rodet, "Globally optimal short-time dynamic time warping, application to score to audio alignment," in *ICASSP 2006 Proceedings*, 2006.
- [8] S. Young, G. Evermann, M. Gales, T. Hain, D. Kershaw, X.A. Liu, G. Moore, J. Odell, D. Ollason, D. Povey, V. Valtchev, and P. Woodland, *The HTK Book (for HTK Version 3.4)*, Cambridge University Engineering Department, 2006.