
Realtime Segmentation and Recognition of Gestures using Hierarchical Markov Models

JULES FRANÇOISE

MASTER'S THESIS
ATIAM 2010-2011

UNIVERSITÉ PIERRE ET MARIE CURIE
IRCAM
TELECOM PARISTECH

IRCAM - INTERACTION MUSICALES TEMPS-RÉEL

Supervisors:

BAPTISTE CARAMIAUX
FRÉDÉRIC BEVILACQUA

baptiste.caramiaux@ircam.fr
frederic.bevilacqua@ircam.fr

Abstract

In this work, we present a realtime system for continuous gesture segmentation and recognition. The model is an extension of the system called *Gesture Follower* developed at Ircam, which is an hybrid model between Dynamic Time Warping and Hidden Markov Models. This previous model allows for a realtime temporal alignment between a template and an input gesture.

Our model extends it by proposing a higher-level structure which models the switching between templates. Taking advantage of a representation as a Dynamic Bayesian Networks, the time complexity of the inference algorithms is reduced from cubic to linear in the length of the observation sequence. We propose various segmentation methods, both offline and realtime.

A quantitative evaluation of the proposed model on accelerometer sensor data provides a comparison with the Segmental Hidden Markov Model, and we discuss several sub-optimal methods for realtime segmentation. Our model reveals able to handle signal distortions due to speed variations in the execution of gestures. Finally, a musical application is outlined in a case study about the segmentation of violin bow strokes.

Résumé

Nous présentons un système temps-réel pour la reconnaissance et la segmentation de gestes continus. Le modèle est une extension d'un système existant appelé *Gesture Follower*. Développé à l'Ircam, ce modèle est un hybride entre le Dynamic Time Warping et les Modèles de Markov Cachés, qui permet l'alignement temporel entre un geste d'entrée et un geste de référence.

Notre modèle étend le précédent par l'ajout d'une structure de plus haut niveau qui modélise les transitions entre gestes de référence. En tirant parti d'une représentation sous forme de réseau bayésien dynamique, la complexité temporelle des algorithmes d'inférence est réduite de cubique à linéaire en fonction de la longueur de la séquence d'entrée. Nous proposons plusieurs méthodes de segmentation, à la fois temps différé et temps réel.

Une évaluation quantitative du modèle est réalisée sur une base de signaux d'accéléromètres, permettant une comparaison avec le modèle de Markov segmental. Par ailleurs, différentes méthodes sous-optimales pour la segmentation temps-réel de geste complexes sont discutées. Notre modèle se révèle adapté à des variations de vitesses lors de l'exécution des gestes qui induisent une distorsion des signaux gestuels. Enfin, une application musicale est présentée au travers d'une étude de cas visant à identifier et segmenter les modes de jeux d'un violoniste.

Acknowledgments

First of all, I would like to thank my supervisors, Baptiste Caramiaux and Frédéric Bevilacqua for accompanying me along this study, for the fruitful discussions and advices, for strengthening my interest in music and gesture.

Overall, I thank them for the strong support to my PhD application, and I am eager to start my thesis in the team.

I thank the whole IMTR team for the good atmosphere and the constructive discussions. Finally, I would like to thank the many people who helped me along this internship – for one reason or another – which include Tommaso and Fivos, H  l  ne, Brett, Thomas, Pierre, Aymeric, Benjamin, J  r  me, Louis, Jose, Julien, and many more.

Contents

1	Introduction	1
2	Background	3
2.1	Gesture	3
2.1.1	Musical interfaces	3
2.1.2	Musical gestures	4
2.1.3	Chunking	5
2.2	Gesture modeling using HMMs	5
2.2.1	Gesture recognition	5
2.2.2	Hidden Markov Models	6
2.3	Multilevel models	8
2.3.1	The Segmental Hidden Markov Model	8
2.3.2	The Hierarchical Hidden Markov Model	9
2.3.3	A unified framework: Dynamic Bayesian Networks	11
2.4	Modeling musical gestures: the <i>Gesture Follower</i>	13
3	Proposed model	15
3.1	Description of the model	15
3.1.1	Introduction	15
3.1.2	Modeling scheme	16
3.2	Efficient implementation using Dynamic Bayesian Networks	17
3.2.1	Representation and formal description	18
3.2.2	Optimal decoding: the Viterbi Algorithm	20
3.2.3	Towards a realtime system: algorithms for approximate segmentation	21
4	Evaluation	25
4.1	Evaluation method	25
4.1.1	Data collection	25
4.1.2	Method	26
4.1.3	Evaluation function	28
4.2	Offline segmentation: a comparison with the segmental model	30
4.2.1	Same subject, same tempo	30
4.2.2	Inter-subject, same tempo	32
4.2.3	Same subject, inter-tempo segmentation	34
4.2.4	Discussion	37
4.3	Towards a realtime system	37
4.3.1	Forward algorithm	37
4.3.2	Fixed-Lag Smoothing	40
4.3.3	Discussion	41
4.4	A musical application: segmentation of violin bowing techniques	41
5	Conclusion and future directions	45
	Bibliography	50

A	Model description: representation and algorithms	51
A.1	Representation	51
A.1.1	Notations	51
A.1.2	Conditional Probability Distributions	52
A.2	Forward Algorithm	53
A.2.1	Forward pass: formalization using the frontier algorithm	53
A.2.2	Reduction	54
A.2.3	Algorithm	55
A.3	Fixed-Lag Smoothing	55
A.4	Viterbi Algorithm	57
A.4.1	Algorithm : Viterbi Decoding	57

List of Figures

1.1	Two projects of the IMTR team	1
2.1	Examples of gestural interfaces for musical expression	4
2.2	Temporal Representation of a HMM	7
2.3	Graphical representation of a SHMM	9
2.4	An example of a simple HHMM	10
2.5	A simple Bayesian network representing the probabilistic dependences between three random variables.	11
2.6	DBN Representation of an input-output HMM	12
2.7	DBN representation of a HHMM with 2 levels of hierarchy	13
2.8	The learning procedure of the <i>Gesture Follower</i>	14
3.1	Segmentation of a complex signal as a sequence of primitive shapes	16
3.2	The topology of the proposed model	17
3.3	The particular learning procedure of the model	18
3.4	DBN representation of the proposed model	19
3.5	Pseudo-code for the Fixed-Lag Smoothing algorithm	23
4.1	Gesture vocabulary of the <i>uWave</i> database	26
4.2	Our gesture database	27
4.3	The two reference segmentations considered in this study	29
4.4	The different types of errors identified by the evaluation function	29
4.5	Acceleration signals of gesture 2 performed at 2 different speeds	32
4.6	Acceleration signals of gesture 2 respectively performed by participants 3 and 7 at 60 bpm	33
4.7	Coarticulation effects	36
4.8	Histogram of the length of the errors (substitutions + insertions) for the Forward algorithm.	39
4.9	A typical example of segmentation computed with the forward algorithm presenting insertion	39
4.10	Fixed-Lag Smoothing: Influence of the lag and comparison with the Forward algorithm	40
4.11	The musical score of the violin phrase, correlated with the audio track and an acceleration signal	42
4.12	Segmentation results on the violin phrase compared for three algorithms	43
A.1	The 2-level HHMM of our study, represented as a DBN. Q_t is the production state at time t , S_t is the symbolic state at time t ; $F_t = 1$ if the sub-HMM has finished (entered its exit state), otherwise $F_t = 0$. Shaded nodes are observed, the remaining nodes are hidden.	51

1

Introduction

At Ircam, the Realtime Musical Interaction Team¹ focuses its activities on interactive musical systems. In particular, current research involve both modeling gesture and sound and developing gesture capture systems and interfaces. Specific applications of the research are defined in the context of performing arts and interaction with digital media.

Two complementary approaches are conducted simultaneously. First, experimental studies aim at analyzing gesture in various contexts, from instrumental playing [Rasamimanana et al., 2009] and dance performance [Bevilacqua and Flety, 2004] to gestural embodiment of environmental sounds [Caramiaux et al., 2011a]. Second, interactive systems for real-time performance are developed, from new musical interfaces (such as the MO shown on figure 1.1(b) [Rasamimanana et al., 2011]) to gesture analysis software. One example of this complementarity is the *Augmented Violin* project, which combined theoretical studies about bowing techniques with the creation of a system for mixed-music allowing for realtime recognition of bowing modes such as *Martelé* and *Détaché*. As capturing instrumentalists' movements was required to be as unintrusive as possible, a specific device was developed displayed on figure 1.1(a).



(a) The augmented violin: example of an instrumented bow



(b) *Modular Musical Objects* (MO)

Figure 1.1: Two projects of the IMTR team

¹IMTR team: Realtime Musical Interactions. <http://imtr.ircam.fr>

Machine learning techniques have been successfully applied to modeling gesture. In particular, Markov models were found suitable for modeling gesture signals as sequential data.

First, a gesture recognition system was developed for the specific context of performing arts, namely the *Gesture Follower* [Bevilacqua et al., 2010]. In order to extract information about the temporal execution of a gesture, the system continuously updates characteristics of the movement.

Second, a recent study investigated a computational model for off-line gesture segmentation and parsing. This work, tested on ancillary gestures [Wanderley et al., 2005], aimed for a quantitative analysis of musical gestures using segment models to highlight their inherent multi-level information content [Caramiaux et al., 2011b].

Our study lies at the intersection between these two approaches. Specifically, the goal of our work is to implement and evaluate a model which extends the *Gesture Follower* by providing a method for gesture analysis following multiple time scales. Based on the Hierarchical Hidden Markov Model (HHMM), the proposed model allows for segmenting complex gestures and gestures sequences based on time profile recognition. Using the formalism of Dynamic Bayesian Networks (DBNs), an efficient implementation is presented and different segmentation methods are proposed, both offline and realtime.

The first chapter is concerned with an overview of the background in gesture modeling. After a general introduction to musical gestures, we review Hidden Markov Models for gesture recognition and study two extensions of the model: the Segmental Hidden Markov Model and the Hierarchical Hidden Markov Model. The models are unified under the general framework of Dynamic Bayesian Networks. The proposed model is detailed in chapter 3 where we expose various segmentation methods, from optimal decoding to sub-optimal methods for realtime segmentation. Finally, we describe a quantitative evaluation of the model in chapter 4. In a first section, the model is compared with the segmental model on offline segmentation. Then, two methods for realtime segmentation are evaluated and discussed to identify a compromise between the accuracy of the segmentation and the delay to realtime.

2

Background

2.1 Gesture

Gesture have been of growing interest in many research fields, bringing an abundance of new definitions. First, it is important to make a distinction between a *movement* – physical action – and a *gesture* which is usually considered as carrying information [Kendon, 2004]. A common definition tends to qualify gestures as body movements, involving for example the head or a hand, which convey meaningful information [Godøy and Leman, 2009].

The computer vision community has come to define the notions of *actions*, considered as simple motion patterns, and *activities* which are complex and involve coordinated actions between humans [Turaga et al., 2008].

Another important role of gesture has been highlighted in the Human-Computer Interaction community (HCI) , namely interacting with the environment [Mitra and Acharya, 2007]. Gestures are then considered as an input modality which aims at controlling and interacting with a computer.

2.1.1 Musical interfaces

Gestures for control, as considered in HCI, may be divided in *manipulative* and *empty-handed* gestures [Godøy and Leman, 2009]. Thus, important developments have been made from new physical interfaces to vision-based gesture recognition systems aiming at interpreting human gestures. In computer music, the development of new interfaces for musical expression dramatically increased over the last decade – even giving its name to an international conference¹.

From the early experiments of Max Mathews and its *Radio baton* (figure 2.1(a)) [Mathews, 1989] to electronic instruments, for example the *Meta-Instrument* [de Laubier and Goudard, 2006], the *T-stick* [Malloch and Wanderley, 2007] or the *Karlax*² (figure 2.1(b)), a great number of musical interfaces have been designed, focusing on gesture inputs for musical expression. This interest in new interfaces for musical expression drives specific research about musical gesture, raising the need of formalization [Cadoz and Wanderley, 2000] and studies of gestural data and gesture-to sound mapping strategies [Wanderley and Battier, 2000].

¹NIME: International Conference on New Interfaces for Musical Expression. <http://www.nime.org>

²Karlax: innovative midi controller developed by *Dafact*. <http://www.dafact.com/>

(a) Max Mathews and the *Radio Baton*(b) The *Karlax*, developed by *Dafact*

Figure 2.1: Examples of gestural interfaces for musical expression

2.1.2 Musical gestures

Music-related gestures are found in a large number of musical activities. Indeed, musical gestures encompass more than controlling musical instruments or coordinating musicians, as, for instance, people tend to make gestures while listening to music: dancing, mimicking instruments, beating rhythms or performing abstract movements.

Recent studies in the field of embodied cognition tend to emphasize the role of gesture in music perception, from listening to performance [Leman, 2006]. An embodied perspective in music suggests that the whole body is involved in our experience of music, and that we are constantly simulating sound-related actions when perceiving or even imagining sound and music. This embodiment process is strongly connected with the multimodal aspect of perception. Relationships between gesture and sound are anchored in our ecological experience of the mechanical coupling between an action and the sound produced [Jensenius, 2007].

A typology of music-related actions has emerged [Cadoz and Wanderley, 2000]. From a functional point of view, we may distinguish *instrumental* or *effective* gestures, involved in the production of sound, from more symbolic gestures such as *accompanying* and *ancillary* gestures which are not involved in the production of sound, but which convey expressivity to the audience. A phenomenological analysis of musical gestures would focus on an analytical description of gestures in terms of *cinematic*, *spatial* and *frequential* characteristics.

This last approach is quite attractive for gesture modeling as it allows for representing gestures in terms of measurable physical parameters. In recent research, quantitative analyses of musical gestures have taken advantage of a representation of gestures as time series, and particularly as time profiles. A major benefit of this assumption is the possibility of comparing quantitatively gesture profiles with sound descriptors to characterize gesture-

sound relationships. For example, experiments on sound-tracing [Godøy et al., 2006] reveal an interesting analogy between gesture morphology and Pierre Schaeffer's *Musical objects* [Schaeffer, 1966], extending his concept to *Gestural-Sonorous objects* [Godøy, 2006]. Recently at Ircam, an experiment has been made to better understand the link between both mental and gestural representations of environmental sounds [Caramiaux et al., 2011a]. Participants were asked to move freely on environmental sounds, putting in evidence two types of behavior. First, sounds which cause is clearly identified often lead participants to mimic the action, whereas they tend to follow salient sound feature profiles when the source cannot be identified.

2.1.3 Chunking

In this framework, if basic categories of gesture-sound objects can be defined, for example following excitatory types (impulsive, sustained, iterative), music performance cannot be reduced to a sequential process. In [Widmer et al., 2003], authors investigate artificial intelligence methods to analyze musical expressivity, highlighting the need for multilevel models: *"Music performance is a multilevel phenomenon, with musical structures and performance patterns at various levels embedded in each other."* If this work focused on analyzing audio and midi data of piano performance, we understand here the importance of considering multiple time scales when studying gestures in the context of musical performance. Recent findings about pianists' finger tapping emphasize two factors constraining musical gestures: biomechanical coupling and *chunking* [Loehr and Palmer, 2007]. Introduced by Miller in the fifties [Miller, 1956], *chunking* suggest that *"perceived action and sound are broken down into a series of chunks in people's mind when they perceive or imagine music"* [Godoy et al., 2010]. More than just segmenting a stream into small entities, chunking refers to their transformation and construction into larger and more significant units. In term of action execution, it involves a hierarchical planning of movements [Rosenbaum et al., 1983]. Thus, studying gesture in an action-sound interaction context should take into account different levels of resolution, organized in a hierarchical structure. This theoretical background opens new perspectives for both analysis and gestural control of music. In the rest of this study, we focus on designing a method able to model gesture according to multiple time scales.

2.2 Gesture modeling using HMMs

2.2.1 Gesture recognition

In the Human-Computer Interaction community, gesture recognition has been of growing interest for the last few years with a wide range of applications such as sign language recognition, navigation in virtual environments, interaction with computers and control of digital media. In HCI, gesture is used as an input device for control, often involving only a few iconic movements to be recognized once completed. Thus, Dynamic Time Warping (DTW) has been widely used in this domain for its ability to model time variations in the execution of a gesture [Corradini, 2001]. However, DTW requires sampling the whole time series, revealing expensive in memory.

Models with hidden states have been introduced to address this limitation, considering their suitable property of compressing information. The basic assumption of these mod-

els is to consider hidden states that generate observations. Evolving in time and possibly given input data, these states aim at modeling an underlying phenomenon which produces observable outputs instead of focusing on the signal itself. The most common examples are Hidden Markov Models (HMMs) and Kalman Filter Models (KFM). In the following, we focus on HMMs and we use this acronym to designate to the standard model defined in [Rabiner, 1989].

First introduced for speech recognition, HMMs revealed efficient for gesture recognition [Yamato et al., 1992]. If many gesture recognition techniques inherit from speech and handwriting recognition, this modality differs from the previous ones in two aspects: data acquisition involves a wide variety of sensing systems, and more important, "*gestures are ambiguous and incompletely specified*" [Mitra and Acharya, 2007]. These problems involve developing domain-specific techniques in order to achieve efficient recognition.

Some shortcomings of HMMs can limit the efficiency of gesture recognition. Indeed, this model is unable to reject unrecognized gestures, deal with geometric invariants – scale, offset, orientation, – and training procedure must be achieved before recognition. In [Lee and Kim, 1999], authors propose a threshold model based on HMMs which allows for the rejection of unrecognized gestures and Eickeler et al introduce filler models in a HMM-based system in [Eickeler et al., 1998]. In order to extract significative gestures in a continuous stream of information, the model proposed in [Bhuyan et al., 2006] focuses on detecting gesture boundaries by identifying "pauses" at the beginning and end of gestures. In [Wilson and Bobick, 1999a], Wilson and Bobick define a model for parameterized gesture recognition and introduce online adaptive learning of gesture models in [Wilson and Bobick, 1999b]. Often, difficulties arise in the interpretation of hidden states. The Semantic Network Model, presented in [Rajko et al., 2007], introduce *semantic states* carrying a symbolic meaning.

2.2.2 Hidden Markov Models

Introduced by [Baum and Petrie, 1966], Hidden Markov Models (HMMs) have been reviewed in the classical tutorial [Rabiner, 1989]. We give here a short description of the Model. For comprehensive studies, refer to [Rabiner, 1989] and [Bilmes, 2006].

Representation

First-order Markov chains model sequential data by considering probabilistic dependences between successive observations. Hidden Markov Models extend this principle by encoding information in hidden states that control the dependence of the current observation on the history of observations.

At each time step, a HMM is characterized by a discrete hidden variable Q_t taking values in a set of N hidden states, and an observation variable Y_t which can be discrete or continuous. Then, three probability distributions are defined:

- ▷ A prior probability distribution $\Pi = \{\pi_i\}$ where $\pi_i = P(Q_1 = i)$ is the probability that the system initializes in state i .
- ▷ A state transition probability matrix $A = \{a_{ij}\}$ where $a_{ij} = P(Q_t = j | Q_{t-1} = i)$ is the probability to make a transition from state i to state j , respecting a first-order Markov property.

- ▷ An observation probability distribution $B = \{b_j(y)\}$ where $b_j(y) = P(Y_t = y | Q_t = j)$ evaluates the probability of observing y given that the hidden variable is in state j .

The behavior of a HMM can be depicted by the temporal representation shown on figure 2.2. The model presents a repeating structure: each time slice contains one hidden node and one observable node. Along this report, we keep the convention to represent hidden states as white circles and observations as shaded nodes. The system first initializes in state S_1 according to Π and emits an observation symbol Y_1 . Then, at each time step a transition is made according to A and the system produces an observation symbol according to B . On the right side of the figure, the parameters are explicitly represented.

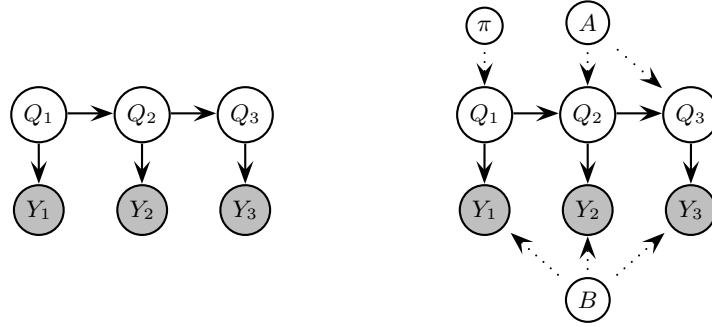


Figure 2.2: The temporal representation of a HMM. The model is unrolled for 3 time slices, but the structure can be repeated for each time step. On the right side, the model parameters A , B and π are explicitly represented.

Inference

In the classical tutorial [Rabiner, 1989], L. Rabiner points out three problems of interest for HMMs:

- ▷ Compute the probability of the sequence $\bar{y} = y_1 \cdots y_T$ given the model.
- ▷ Find the sequence of hidden states $\bar{q} = q_1 \cdots q_T$ which best explains the observation sequence.
- ▷ Adjust model parameters $\lambda = (A, B, \pi)$ which maximizes the probability of the sequence.

The Forward-Backward algorithm has been introduced to solve the first problem. The basic principle is to define the forward variable $\alpha_t(i) = P(y_1 \cdots y_t, Q_t = i | \lambda)$. Initialized to $\alpha_1(i) = \pi_i b_i(y_1)$, the variable is then updated at each new observation by summing over the possible transitions:

$$\alpha_{t+1}(j) = \left[\sum_{i=1}^N \alpha_t(i) \cdot a_{ij} \right] \cdot b_j(y_{t+1}) \quad (2.1)$$

The most probable state at time t can be estimated as the index maximizing α_t . Finding the optimal sequence of hidden states given the observation sequence is solved using the Viterbi algorithm. First, a variable δ is updated at each time step given a formula similar to equation 2.1, changing the sums to maximizations and keeping track of the arguments which maximized δ . Then, the optimal state sequence is computed by a backtracking operation.

Limitations

If HMMs are extensively used in recognition systems, some limitations of the model can reveal problematic in various contexts [Ostendorf et al., 1996]. First, the probability of staying a duration τ in the same state derives from the auto-transition probability of a state: $p(\tau) = a_{ii}^{\tau-1}(1 - a_{ii})$. As a consequence, the state duration modeling suffers from an implicit definition and decreases exponentially. Second, feature extraction is conditioned by the production of observations at a frame level. As a consequence, the standard model is too weak for a segmentation task which requires higher-level modeling.

2.3 Multilevel models

2.3.1 The Segmental Hidden Markov Model

In speech recognition, segment models were proposed to overcome the restriction on feature extraction imposed by frame-based observations. In the Segmental Hidden Markov Model (SHMM), each hidden states emits a sub-sequence of observations rather than a single one, given a geometric shape and a duration distribution. Successfully applied to speech recognition [Ostendorf et al., 1996], handwritten shape recognition [Artieres et al., 2007] and, at Ircam, time profile recognition of pitch and loudness [Bloit et al., 2010]; the Segmental HMM was used in a recent study for gesture modeling [Caramiaux et al., 2011b]. The model aimed at segmenting the gestures of a clarinetist. The model was used to represent a continuous stream of gestures as a sequence of geometric shapes extracted from a given dictionary. Tested on ancillary gestures, the model provided a quantitative analysis of the performance of a musician, highlighting recurrent patterns of ancillary gestures correlated to the musical performance.

In this section, we give a brief overview of the Segmental HMM, an extensive study of the general model is formalized in [Ostendorf et al., 1996].

Representation

The SHMM extends the standard HMM by defining the observation probability distribution at a segment level. Instead of emitting a single symbol, a hidden state produces a variable-length sequence of observations. Let's note $y_{t_1:t_2} = [y_{t_1} \cdots y_{t_2}]$ a subsequence of observations of duration $l = t_2 - t_1 + 1$. In a standard HMM, the observation probability is defined at a sample level:

$$b_j(y) = P(Y_t = y | Q_t = j)$$

In a SHMM, a hidden states emits a sequence of observations of length l given:

$$\begin{aligned} P(y_{t_1}^{t_2}, L_t = l | Q_t = j) &= P(y_{t_1}^{t_2} | Q_t = j, L_t = l) \cdot P(L_t = l | Q_t = j) \\ &= b_{j,l}(y_{t_1}^{t_2}) \cdot P(L_t = l | Q_t = j) \end{aligned}$$

where $b_{j,l}(y_{t_1}^{t_2})$ is the probability of the subsequence of observation given the base shape and the duration l and $P(L_t = l | Q_t = j)$ is the probability of having a duration l given that the system is in state j . As a consequence, a new distribution is introduced to define the possible durations of the segments. A graphical representation of a SHMM is shown on figure 2.3.

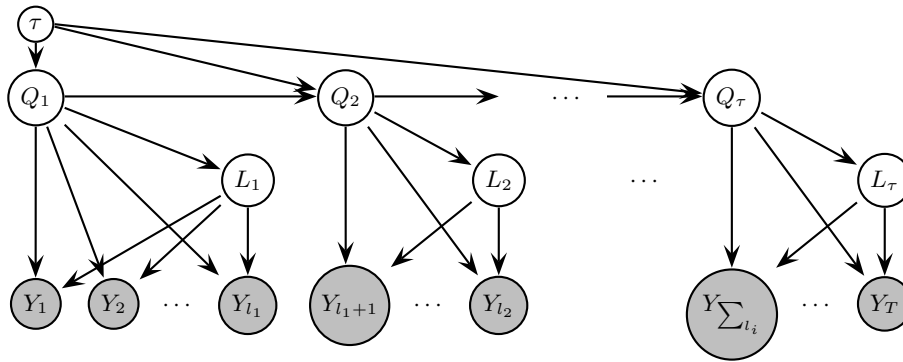


Figure 2.3: Graphical representation of a SHMM

Inference

When using the SHMM, the first step is to define a dictionary of primitive shapes. These can be defined given prior knowledge – for example considering simple geometric shapes – or can be learned during a training phase. A simple technique is to annotate manually a reference signal to extract short segments to feed the model with a single template.

Similarly to HMMs, three problems can be defined for segment models. Here, each internal state generates a duration and a subsequence of observations. So finding the optimal sequence of hidden states and durations is equivalent to find the best representation of the signal as a sequence of segments characterized by a label – the geometric shape – and a duration.

This task is achieved by a dynamic programming algorithm analogous to the Viterbi algorithm of HMMs. As the duration distribution appears like a new dimension, the complexity of the decoding algorithm increases and the optimal decoding algorithm is called the 3D Viterbi algorithm.

Limitations

If some of the limitations of HMMs have been solved introducing the Segmental HMM, the model is only able to handle one level of hierarchy governing the transitions between segments, thus limiting the analysis to this unique time scale. Moreover, gestures and particularly musical gestures are subject to timing variations. Different executions of the same gestures reveal local speed variations involving a non uniform time stretching of the primitive shapes composing a gesture. In the SHMM, the decoding process amounts to fit the geometric shapes to an input signal by applying a uniform scale transformation to the primitive shapes. This implies that the only transformation of the signal allowed is a uniform time-stretch over a whole segment which is not suitable for our application. The Hierarchical HMM defined in the next section overcomes these restrictions, allowing an arbitrary number of hierarchy levels and modeling segments by Markov process instead of geometric shapes.

2.3.2 The Hierarchical Hidden Markov Model

The Hierarchical Hidden Markov Model (HHMM) [Fine et al., 1998] extends the standard HMM by making each state an autonomous model, allowing a deep hierarchy. The model has been applied to speech recognition to model different imbricated time scales, from phonemes to words and sentences.

Representation

The Hierarchical Hidden Markov Model (HHMM) extends the standard HMM by making each of the hidden state an "autonomous model". Hidden states of the model are classified in two classes:

- ▷ *production states*: states which emit observations similarly to the hidden states of a HMM.
- ▷ *internal states*: instead of emitting observation, an internal state generates an autonomous model.

Thus, each internal state produces a sequence of observations by recursively activating its substates until a production state is reached. When the system has finished at the current level, it reaches an *exit* state which allows to go back to the parent node and make a higher level transition.

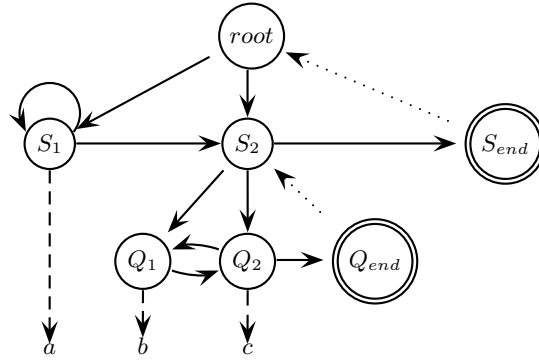


Figure 2.4: An example of a simple HHMM

In order to better understand the behavior of such a model, consider the simple HHMM of figure 2.4. The model has two levels of hierarchy and generates symbols associated with letters. Starting from the root a time $t = 0$, the model will make a *vertical* transition at $t = 1$ according to a prior probability distribution to enter a state of the first level, for example S_1 . Then, the model will emit the symbol "a" and make a transition, either to the same state or to S_2 . As S_2 is an internal state, the system will make a vertical transition and enter for example Q_1 . Once produced the symbol "b" the system will make a transition to Q_2 and emit "c". After looping between Q_1 and Q_2 , the system will reach an *exit* state – namely Q_{end} – and go back to the parent state S_2 to make a transition at the first level. Here the only possibility is to reach S_{end} and go back to root. Finally, the system would generate the regular expression $a^x(bc)^y$. More importantly, we can notice that the internal state S_2 would generate the subsequence $(bc)^y$. So, this structures inherently handles a signal analysis on multiple time scales.

Inference

Introduced as an extension of the standard HMM, this model have been studied in a similar way, defining the three classical problems. Considering the recursive structure of the model, each internal state calling a submodel, Fine et al. proposed recursive algorithms for the Hierarchical HMM [Fine et al., 1998], inspired from the *Inside-Outside* algorithm for Stochastic Context-Free Grammars. Notably, an equivalent to the Viterbi algorithm was designed to compute the optimal sequence of hidden states for each level. Unfortunately,

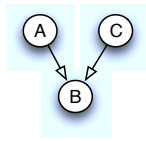


Figure 2.5: A simple Bayesian network representing the probabilistic dependences between three random variables.

the time complexity of the algorithm is cubic in the length of the observation sequence, which becomes intractable even for time series analysis. In the next section we introduce Dynamic Bayesian Networks which provide powerful algorithms for Markov models.

2.3.3 A unified framework: Dynamic Bayesian Networks

The inability of HMMs to deal with multiple time scales led researcher in activity recognition to develop more complex models showing a hierarchical structure. Often based on computer vision techniques, activity recognition knew a growing interest in the last few years with applications to surveillance, behavioral biometrics and interactive environments [Turaga et al., 2008]. Activities, defined as complex sequences of primitive actions, require multi level models, from feature extraction and action recognition to high level schemes for activity recognition. Recently, Dynamic Bayesian Networks (DBNs) raised attention considering their ability to encode complex conditional dependences in dynamic systems, thus permitting hierarchical modeling of human activities [Subramanya et al., 2007]. As shown by K. Murphy [Murphy, 2002], this framework allows the expression of every Markov model, offering a simple representation and powerful inference algorithms.

Bringing together graph theory and probability theory, graphical models provide a powerful formalism for statistical modeling [Wainwright and Jordan, 2007]. In a graphical model, a set of nodes representing random variables are connected together by directed or undirected arcs defining probabilistic dependences between these variables. Bayesian networks are the family of graphical models with directed arcs. In such models, a directed arc between two nodes defines a conditional probability distribution between two random variables – a *parent* node and its *child*, – offering both a simple representation and strong stochastic modeling of causality.

A simple example is given on figure 2.5 where three random variables are represented. The edges indicate a causal relationship between different events, for example $A \rightarrow B$ means that we can evaluate the probability of observing B given knowledge about A. The scheme also shows that A and C are not independent given B, meaning that the inference we can make about A given B is conditioned on the state of C.

Specifically developed to model dynamic systems, Dynamic Bayesian Networks (DBNs) are a special case of Bayesian networks which contain edges pointing in the direction of time. In fact, the graphical representation of a DBN shows a recurrent scheme in which a unique structure is repeated at each time slice. This structure contains an arbitrary number of connected nodes, a part of which have children in the following time slice.

Dynamic Bayesian Networks can be seen as an extension of Hidden Markov models with an arbitrary number of hidden and observable states per time slice, representing complex dependences. The simplest DBN is in fact the HMM, which only contains a hidden state

and an observation per time slice. Representing a HMM as a DBN only lies on unrolling its temporal representation, as presented on figure 2.2 of section 2.2.2.

Representation

We give now a short formal description of DBNs. For an extensive study, see [Murphy, 2002].

A DBN is a directed acyclic graph, where a set of nodes defines random variables Z_t , which can be partitioned in *hidden*, *observable* and *input* variables: $Z_t = (X_t, Y_t, U_t)$. A DBN is characterized by a prior $P(Z_1)$ and a two-slice Temporal Bayes Net (2TBN) which defines $P(Z_t|Z_{t-1})$ as follows:

$$P(Z_t|Z_{t-1}) = \prod_{i=1}^N P(Z_t^i | Pa(Z_t^i))$$

where Z_t^i is the i 'th node at time t , which could be a component of X_t , Y_t or U_t , and $Pa(Z_t^i)$ are the parents of Z_t^i in the graph, which can either belong to the same or the previous time slice.

A simple example is given on figure 2.6 where an Input-Output HMM is represented as a DBN. The unit structure contains 3 nodes, one for each possible type. Only hidden nodes have children in the following time slice. The model is characterized by a prior on the first time slice, and by the 2TBN represented by time slices 2 and 3 which is sufficient to encode the whole temporal structure of the model.

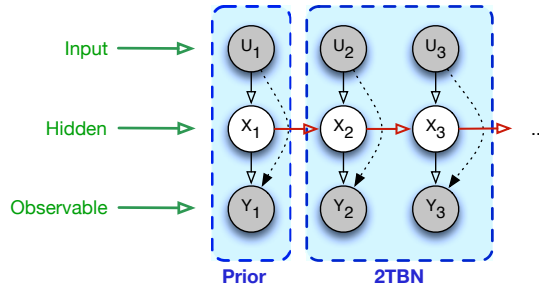


Figure 2.6: An input-output HMM represented as a DBN. The unit structure of the model contains one input node, one hidden node and one observation. Red arrows represent temporal relationships.

In addition to offering a simple temporal representation, DBNs allows for deriving powerful inference algorithms, from exact inference with the Frontier algorithm [Zweig, 1996], to approximate methods, for example particle filtering [Doucet et al., 2000].

Expressing the HHMM as a DBN

The Hierarchical HMM detailed in the previous section can be expressed as a DBN. On figure A.1, a HHMM with two levels of hierarchy is represented as a DBN. In each time slice, an internal state S_t and a production state Q_t are connected together and have children in the next time slice. Two binary nodes are introduced to handle the *exit* states. For each level of the hierarchy, these nodes will activate if the sub-HMM has entered its *exit* state, enabling the system to make a transition at the parent level. This representation as a DBN allows for deriving powerful inference algorithms. Particularly, we focus on the *Frontier* algorithm, generalization of the Forward-backward algorithm, which time

complexity is linear in the length of the observation sequence. As the proposed model detailed in the next chapter is a special case of HHMM, we don't give here a comprehensive description of the model. Representation and algorithms are detailed for our specific model in order to lighten the notation.

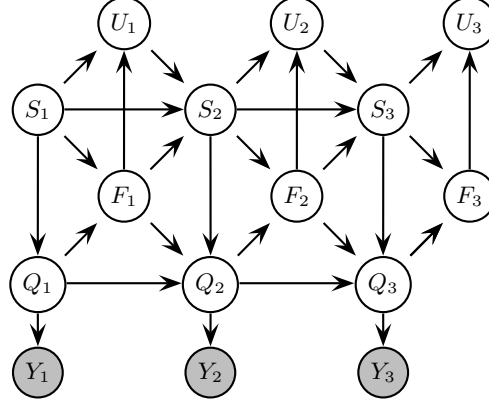


Figure 2.7: DBN representation of a HHMM with 2 levels of hierarchy. Q_t is the production state at time t , S_t is the internal state at time t ; $F_t = 1$ if the sub-HMM has finished (entered its exit state), otherwise $F_t = 0$. Shaded nodes are observed, the remaining nodes are hidden.

2.4 Modeling musical gestures: the *Gesture Follower*

Often, online gesture recognition systems tend to define gestures as unbreakable units which can only be recognized once completed. In music and performing arts, the way a gesture is performed is often more important than identifying the gesture itself. Considering the specific constraints of this context, a realtime system for gesture-based interaction was developed in the IMTR team. A novel approach was proposed: instead of recognizing discrete gesture units, the system continuously updates a set of parameters characterizing the execution of a gesture. Particularly, the system allows for a realtime estimation of the likelihood of a gesture and the *time progression*, answering the question: "*Where are we within the gesture ?*".

The basic assumption is that gestures can be represented as multidimensional temporal curves. In order to reach a precise temporal analysis, authors focus on modeling time profiles at a sample level using HMMs with a particular topology. We can consider the model as an hybrid scheme between HMMs and Dynamic Time Warping (DTW). We now briefly review the modeling scheme of the system, for a complete review, see [Bevilacqua et al., 2010]. In the rest of this report, we will refer to this model by "*Gesture Follower*".

Representation

To fit the constraints imposed by the context of performing arts for which the set of examples cannot be extensive, a particular learning procedure was adopted, illustrated on figure 2.8. A single example is used as a template to build a Left-Right HMM, in which each state is associated with a sample of the reference gesture. The observation probability is defined as a Gaussian distribution centered around the sample of the reference. The standard deviation of this normal distribution corresponds to the variability between different performances of the same gesture and need to be set a priori or given preliminary

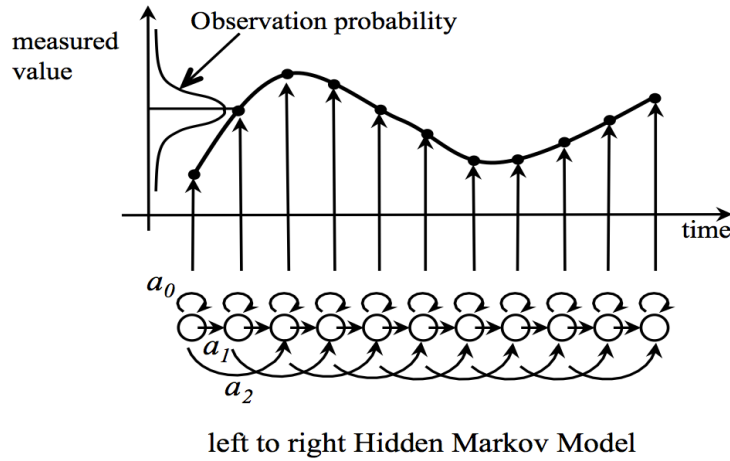


Figure 2.8: The learning procedure of the *Gesture Follower*. A left-right HMM is build from one example curve.

experiments.

Considering its left-right topology, the system models precisely the temporal structure of the gesture and is able to handle time variations in its execution. In addition, assumptions are made about the transition probabilities. On figure 2.8, transitions a_0 , a_1 and a_2 stand for *self*, *next* and *skip* probabilities, which can be set in according to the expected variability of the speed of execution. For example, defining $a_0 = a_1 = a_2 = 1/3$ allows for performing the gesture twice faster or slower than the original example, with equal probabilities of speeding up or down.

decoding

With respect to the constraint of realtime, the Viterbi algorithm typically used to decode the optimal sequence of hidden states is excluded. Here, the standard forward procedure for HMMs is preferred (see section 2.2.2). At each time step, an update of the forward variable α estimates the likelihood of the gesture given a partial observation sequence. Moreover, the time progression can be derived from the index of the state maximizing the forward variable at each time step:

$$time\ progression\ index(t) = \underset{i}{\operatorname{argmax}} \alpha_t(i)$$

Applications

The system, implemented as an external object for Max/MSP³, was called *Gesture Follower*. Among applications, a time synchronization paradigm has emerged which allows for controlling the playback of recorded sounds, for example conducting a virtual orchestra.

³Max/MSP/Jitter: visual programming language for music and multimedia. <http://cycling74.com/>

3

Proposed model

To remind the reader, we are interested here in designing a model which aims at analyzing gestures following multiple time scales, extending the hybrid model called *Gesture Follower*. Using the formalism of Hierarchical HMMs, the proposed model provides a multilevel representation of gestures which goal is the segmentation of complex gestures and gesture sequences.

The chapter begins with a formal description of the model. Then, using the formalism of Dynamic Bayesian Networks, an effective implementation is detailed and different segmentation algorithms are exposed, from exact decoding to sub-optimal methods for realtime segmentation.

3.1 Description of the model

3.1.1 Introduction

The proposed model represents a gesture as a sequence of primitive shapes modeled by Markov chains. An illustration of our goal is shown on figure 3.1. We can imagine capturing a gesture using a single axis accelerometer, thus representing the gesture as a unidimensional temporal curve. Once defined three primitive gestures (a) – *deceleration*, *acceleration* and *constant speed* –, the model must be able to segment a target signal (b) as a sequence of these primitive shapes (c). For the example of figure 3.1, the segmentation would output the series of symbols 12321, together with their temporal alignment with the reference.

As illustrated by this simple example, two challenging issues have to be solved:

1. find the correct sequence of primitive shapes, as a series of symbols
2. detect the time boundaries of these shapes to infer a correct temporal alignment of segments

A desirable property of the model is the ability to handle time variations within segments: the same template must be able to fit a signal with important time variations. For example, the first and last segment of the curve of figure 3.1(c) are derived from the same template, but present different temporal unfolding. Moreover, as for the *Gesture Follower*, the specific context of performing arts requires modeling fine temporal variations, requiring a precision at a sample level.

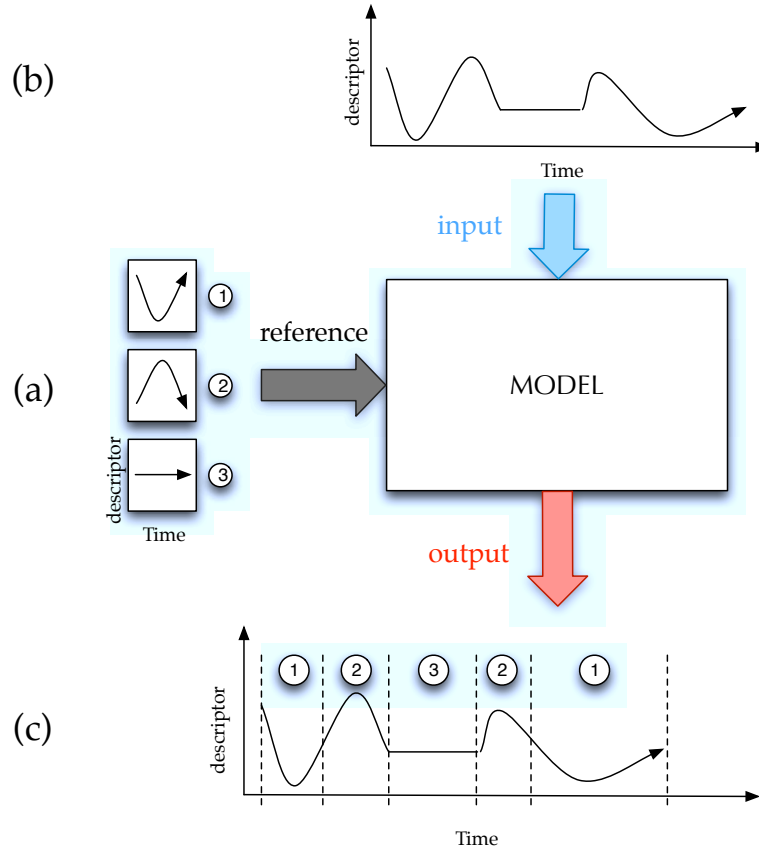


Figure 3.1: Segmentation of a complex signal as a sequence of primitive shapes. A gesture is represented by a time profile of a descriptor, for example acceleration. 3 primitive gestures are learned in the model (a). Segmenting the input signal (b) corresponds to identifying the correct sequence of primitive shapes and their temporal alignment (c).

3.1.2 Modeling scheme

We propose a model, special case of the Hierarchical HMM defined in [Fine et al., 1998], with two levels of hierarchy. A graphical representation of the topology of the model can be found on figure 3.2.

The first level s_i of the hierarchy contains a small number of internal states which are respectively associated with primitive gestures. On the figure, only two internal states s_1 and s_2 are illustrated, each standing for a primitive shape. As this level models the transitions between gestural segments, it will be called *symbolic* level in the following. As a first approximation, no assumption is made about gesture sequences and this high level structure is considered ergodic. However, high level transitions can be set given a priori knowledge, or could be learned from an extensive set of gesture sequences. Each of these *symbolic* states generates a submodel which encodes the temporal evolution of the corresponding primitive gesture. This submodel inherits from the particular topology of the hybrid model of section 2.4 called *Gesture Follower*, associating each sample of a reference gesture to a state in a left-right Markov chain. Hence, observations are only emitted at this second level of the hierarchy – the *production* level. As it focuses on a time scale at a sample level, it will then be named *signal* level.

As for the *Gesture Follower*, a simplified learning procedure is adopted, illustrated on figure

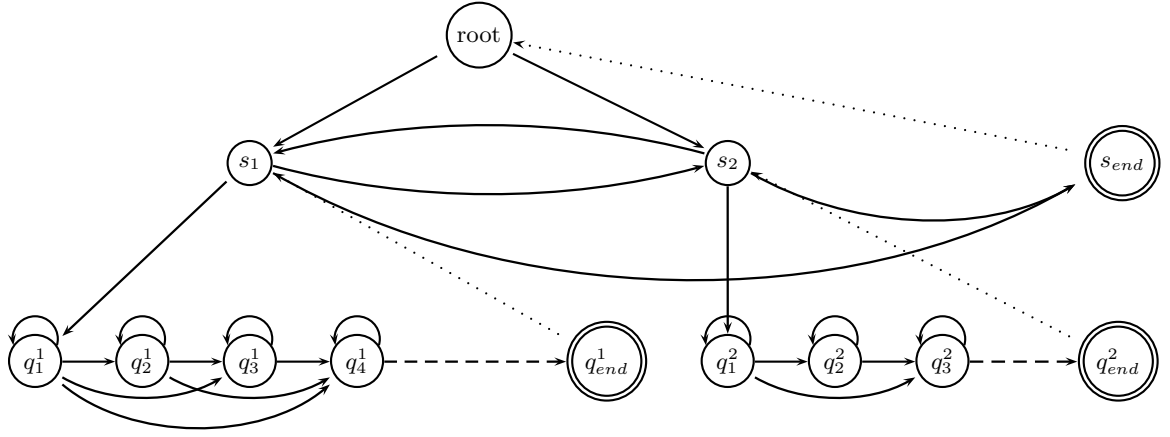


Figure 3.2: The topology of the proposed model. Only 2 symbolic states are represented. *exit* states can be reached if the gesture is close to complete, in order to go back to the parent node and make a transition at the symbolic level

3.3. A single example is needed during the learning procedure. This template gesture is cut given a priori knowledge to constitute a dictionary of primitive shapes. Each sample of a primitive shape is then associated with a state of the *signal* level to form a submodel, which has a left-right topology. The transition probabilities of this Markov chain are fixed a priori given the expected variability in the timing of execution. As each state is a sample, the transition probabilities define the time-stretching allowed. For example, auto-transitions permit to slow down and skipping states amounts to speeding up within the time profile. The prior probability for submodels is a equal to 1 for the first state and zero elsewhere, ensuring that a segment can only be entered from its first sample. The probabilities of reaching an *exit* state are equal to zero except for a few samples at the end of the gesture, ensuring that a transition between two segment is only possible if the current primitive shape is about to finish.

As we associated each *symbolic* state with a primitive shape we are interested in finding the correct sequence of *symbolic* states and the instants of the transitions to obtain the segmentation.

3.2 Efficient implementation using Dynamic Bayesian Networks

In [Fine et al., 1998], authors propose a recursive algorithm which generalizes the Viterbi algorithm of HMMs [Rabiner, 1989]. If this algorithm computes an optimal decoding of the sequence of internal states, offering a precise segmentation technique, its computational complexity strongly limits a use with our model. Precisely, the time complexity of the Viterbi algorithm is cubic in the length of the observation sequence. Considering the particular topology of our model, where each state of the *signal* level is associated with a sample of the reference time profile, this algorithm reveals intractable, even for short sequence.

Taking advantage of representing the HHMM as a Dynamic Bayesian Network, powerful inference algorithm can be derived, reducing the complexity from cubic to linear in the length of the observation sequence. In this section we introduce the representation of our

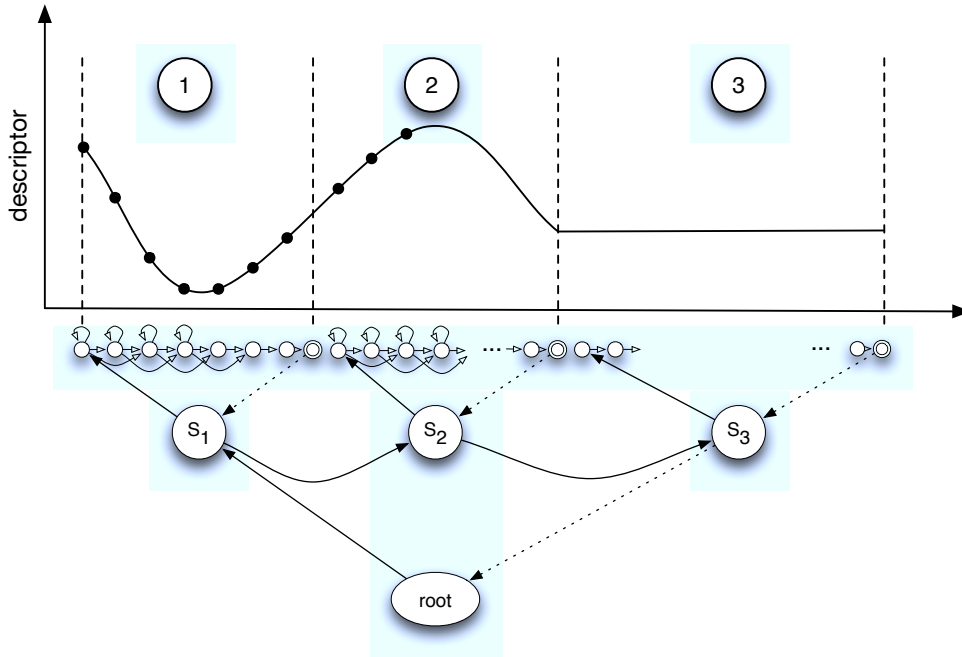


Figure 3.3: The particular learning procedure of the model. Each sample of a reference primitive shape is associated with a state of a sub-HMM, itself generated by an *symbolic* state associated with a segment.

model as a DBN and detail its formal description. Then different segmentation algorithms are proposed: the Viterbi algorithm which performs an optimal decoding; and sub-optimal methods for realtime segmentation.

3.2.1 Representation and formal description

Dynamic Bayesian Networks (DBNs) have been quickly introduced in section 2.3.3. DBNs are a generalization of HMMs allowing an arbitrary number of nodes per time slice to model complex dependences. Notably, Hierarchical HMMs can be represented as DBNs. Figure 3.4 depicts the graphical representation of the DBN corresponding to the two-levels HHMM of our study.

The model is characterized by a set of M *symbolic* states associated with primitive gestures. Each internal state i generates a sub-HMM of length $M^{(i)}$ – length in samples of the reference segment. This can be represented at each time step by considering two random variables respectively associated with the *symbolic* state and the *production* state at time $t \in \llbracket 1; T \rrbracket$. In each time slice, 4 hidden states are defined: a *symbolic* state S_t , a *production* state Q_t , and two binary variables F_t and U_t . These binary nodes are introduced to handle *exit* states characteristic of the Hierarchical HMM. F_t will turn to 1 if the segment has finished, i.e. is about to enter its *exit* state, and U_t will turn on if the entire gesture has finished.

Five probability distribution are necessary to complete the model:

▷ $H = \{h_i\}$: Prior probabilities for the *symbolic* level

$$h_i = P(S_1 = i) \quad i \in \llbracket 1; M \rrbracket$$

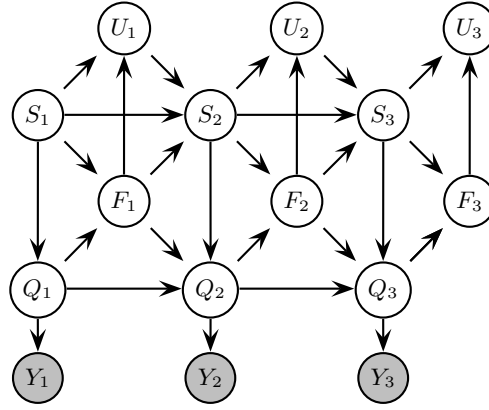


Figure 3.4: DBN representation of the proposed model, HHMM with 2 levels of hierarchy. Q_t is the production state at time t , S_t is the internal state at time t ; $F_t = 1$ if the sub-HMM has finished (entered its exit state), otherwise $F_t = 0$. Shaded nodes are observed, the remaining nodes are hidden.

▷ $G = (g_{il})$: State transition probability matrix for the *symbolic* level

$$g_{il} = P(S_{t+1} = l | S_t = i) \quad i, l \in \llbracket 1; M \rrbracket$$

▷ $\Pi^{(i)} = \{\pi_j^{(i)}\}$: Prior probability distribution for primitive i (vertical transition probability)

$$\pi_j^{(i)} = P(Q_t = j | S_t = i) \quad i \in \llbracket 1; M \rrbracket; j \in \llbracket 1; M^{(i)} \rrbracket$$

▷ $A^{(i)} = (a_{jk}^{(i)})$: State transition probability within primitive i

$$a_{jk}^{(i)} = P(Q_{t+1} = k | Q_t = j, S_t = i) \quad i \in \llbracket 1; M \rrbracket; j, k \in \llbracket 1; M^{(i)} \rrbracket$$

▷ $B^{(i)}(y_t) = \{b_j^{(i)}(y_t)\}$: Emission probability distribution

$$b_j^{(i)}(y_t) = P(Y_t | Q_t = j, S_t = i) \quad i \in \llbracket 1; M \rrbracket, j \in \llbracket 1; M^{(i)} \rrbracket$$

As for the *gesture Follower*, the observation probability is defined as a gaussian distribution:

$$b_j^{(i)}(y_t) = \frac{1}{\sigma\sqrt{2\pi}} \cdot \exp \left[-\frac{\|y_t - \mu_j^{(i)}\|^2}{2\sigma^2} \right]$$

where y_t is the observation at time t and $\mu_j^{(i)}$ is the j^{th} sample of the primitive gesture i . The parameter σ , standard deviation of the normal distribution is considered constant on a whole gesture and has to be set given prior knowledge on the variability between various executions of the same gesture.

The model is finally parametrized by a set of parameters:

$$\lambda = \left\{ H, G, \left\{ \Pi^{(i)} \right\}_{i \in \llbracket 1; M \rrbracket}, \left\{ A^{(i)} \right\}_{i \in \llbracket 1; M \rrbracket}, \left\{ B^{(i)} \right\}_{i \in \llbracket 1; M \rrbracket} \right\}$$

The complete description of the conditional probability distributions of the model is given in appendix A.1. We remind that *exit* states need to be reached in order to go back to the parent level and make a transition between two *symbolic* state, i.e. between two segments. To clarify the role of the binary variables introduced to handle *exit* state, we define the probabilities of these nodes to activate as:

$$P(F_t = 1 | Q_t = j, S_t = i) = a_{j \text{ end}}^{(i)}$$

$$P(U_t = 1 | S_t = i, F_t = f) = \begin{cases} 0 & \text{if } f = 0 \\ g_{i \text{ end}} & \text{if } f = 1 \end{cases}$$

where $a_{j \text{ end}}^{(i)}$ is the probability of reaching an *exit* state from state j of primitive i ; and $g_{i \text{ end}}$ is the probability of reaching a high level *exit* state from primitive i . The binary variable U_t which characterizes the possibility of terminating a gesture, is conditioned by the termination of the current primitive shape, hence the dependence to the value of F_t . To force all segmentations to be consistent with the length of the sequence, we must ensure that all sub-hmms have reached their final state, assuming $U_T = 1$, and $F_T = 1$.

3.2.2 Optimal decoding: the Viterbi Algorithm

This temporal representation of Hierarchical HMMs allows for deriving powerful inference algorithms. Here, we are particularly interested in exact inference, but many approximate techniques have been developed in a general case by K. Murphy. A very complete tutorial about Dynamic Bayesian Networks can be found in [Murphy, 2002].

Principle and definitions

For HMMs, the forward-backward algorithm is based on an update of the probability distributions between two time steps, giving that the hidden node X_t separates the past from the future. Here, 4 hidden nodes are defined in each time slice. The *Frontier* algorithm proposed in [Zweig, 1996] is a way of updating the joint probability over a set of nodes, without needing to create a *Macro-node* $X_t = \{S_t, Q_t, F_t, U_t\}$ which would require a large transition matrix. The basic idea is to create a *frontier* containing at time t all the nodes of the time slice. Then, the frontier is extended by progressively including children nodes from the next time slice and marginalizing over parent nodes to exclude them from the frontier. This operation is repeated until the frontier only contains the nodes of time slice $t + 1$, meaning that the joint probability over the set of nodes has been updated. A backward pass can be defined in a similar manner.

This procedure permits the derivation of an efficient generalization the Viterbi algorithm. Define the δ variable:

$$\delta_t(j, i, f, u) = \max_{X_{1:t-1}} P(Q_t = j, Q_{1:t-1}, S_t = i, S_{1:t-1}, F_t = f, F_{1:t-1}, U_t = u, U_{1:t-1}, y_{1:t})$$

The update process is explicated in appendix A.4, and leads to the following recurrence relation:

$$\begin{aligned}
\delta_t(Q_t, S_t, F_t, U_t) &= P(y_t|Q_t, S_t) \cdot P(U_t|S_t, F_t) \\
&\quad \cdot P(F_t|Q_t, S_t) \\
&\quad \cdot \max_{Q_{t-1}, F_{t-1}} \{P(Q_t|Q_{t-1}, F_{t-1}, S_t)\} \\
&\quad \cdot \max_{S_{t-1}, U_{t-1}} \{P(S_t|S_{t-1}, U_{t-1}, F_{t-1})\} \\
&\quad \delta_{t-1}(Q_{t-1}, S_{t-1}, F_{t-1}, U_{t-1}) \}
\end{aligned}$$

This expression can seem quite complex but important simplifications can be done considering that F_t and U_t are binary variables. In fact, the couple $\{F_t; U_t\}$ can take 3 possible values, because the whole gesture can finish only if the current primitive shape has terminated. Introducing $E_t = F_t + U_t$ ($E_t \in \{0, 1, 2\}$), we propose a new definition of the dynamic programming variable:

$$\delta_t^e(k, l) = \max_{X_{1:t-1}} \log P(Q_t = k, S_t = l, E_t = e, Q_{1:t-1}, S_{1:t-1}, E_{1:t-1})$$

A simplified algorithm can then be computed separating the 3 different cases.

As for HMMs, the indices maximizing δ are kept in order to retrieve the sequence of hidden states in the backtracking operation. Finally, the optimal state sequence is obtained, giving at each time step the *symbolic* state, i.e. the most probable primitive gesture and the *production* state which provides the time progression in comparison with the original template.

Complexity

Due to its recursive nature, the time complexity of the original Viterbi algorithm is $O(KMT^3)$, where M is the number of primitive gestures of mean length K and T is the length of the observation sequence [Fine et al., 1998]. Moreover, the memory complexity is in $O(KMT^2)$.

Here, a short analysis of the algorithm shows that its time complexity is in $O((KM)^2T)$. So, the time complexity is now linear in the length of the observation whereas it was cubic with the recursive algorithm. However, this gain is done at the expense of the space complexity, which is now quadratic in the total number of states of the model. Another important gain is the memory: since the algorithm just involves an update, the most consuming operation is stocking the indices in the matrix Ψ . This requires $O(KMT)$ which is an important reduction compared with the original algorithm, quadratic in the length of the observation sequence.

If this algorithm provides an optimal segmentation of a gesture given a dictionary of primitive shapes, it is inappropriate for designing a realtime system because the whole observation sequence is required. In the next section different methods based on the *Frontier algorithm* are proposed to offer approximate solutions to the segmentation problem in order to move towards a realtime system.

3.2.3 Towards a realtime system: algorithms for approximate segmentation

In the hybrid model called *Gesture Follower*, a simple forward pass is used to estimate at each time step the most probable state, giving access either to the likelihood of the

gesture and the time progression within this gesture. In the following, two procedures are presented: a causal algorithm which consists on a forward pass similar to the method used in the *gesture Follower*, and Fixed-Lag Smoothing which includes a backward pass a few samples, improving the accuracy of the simple forward algorithm but delaying the results of a duration equal to the lag.

Forward algorithm

Filtering corresponds to computing $p(X_t|y_{1:t})$, which allows to perform a realtime segmentation in the sense that the most probable *symbolic* and *production* states are updated at each time step. The algorithm is just the forward procedure of the *frontier* algorithm.

We first need to define the Forward variable as the probability of being at time t in state k of primitive l given the partial observation sequence until time t :

$$\alpha_t^e(k, l) = P(Q_t = k, S_t = l, E_t = e, y_{1:t})$$

As it derives from the same general procedure, the algorithm is similar to the forward pass of the Viterbi algorithm, replacing maximizations by sums. For a complete derivation of the algorithm, the reader can refer to appendix A.2.

The most probable symbolic state at each time step can be computed by:

$$\{Q_t^*, S_t^*, E_t^{(*)}\} = \underset{k, l, e}{\operatorname{argmax}} \alpha_t^e(k, l)$$

Hence, an approximate segmentation can be obtained. An major difference with the Viterbi algorithm is to be noticed: here the most probable state is computed for each time step, whereas it is the most probable sequence in the Viterbi algorithm. As a consequence, the resulting sequence of hidden states may not be consistent as the local maximization does not impose to respect state transitions.

As the algorithm is quite similar to the Viterbi algorithm, the computational complexity of the forward algorithm is linear in the length of the observation sequence: $O((KM)^2T)$.

Fixed-Lag Smoothing

Fixed-Lag Smoothing (FLS) is the process of estimating a state of the past given the partial observation sequence up to the current time, i.e. computing $p(X_{t-L}|y_{1:t})$. This algorithm again derives from the *frontier* algorithm. In addition to the forward update introduced before, a backward pass has to be computed. As the backward update is built similarly to the forward update, it is not detailed here (see appendix A.3).

The algorithm, written in pseudo code on figure 3.5, outputs at each time step t the most probable state at time $t - L$ given the evidence up to time t , where L is a constant called lag. At each time step, a forward update is computed. Then, after initializing the backward variable to 1, a backward update is iterated from time t to $t - L$. Finally, the most probable state at time $t - L$ is defined as the index which maximizes $\gamma_{t-L} = \alpha_{t-L} * \beta_{t-L}$.

It is evident that compared to the filtering algorithm, the number of operation is increased by the addition of a backward pass. In fact the time complexity of the algorithm is in $O(K^2M(M + L)T)$. However, as the estimation of the state at time t depends on future events, the quality of the segmentation is expected to be better than with the


```

t = 1:
    alpha[1] = init_forw(y[1])

FOR t = 2:∞
    alpha[t] = forward(y[t] , alpha[t-1])
    IF t >= L

        beta = 1
        FOR k = 0:L DO
            beta = backward(y[t-k] , beta)
        END FOR
        gamma[t-k] = alpha[t-k].*beta
        state[t-k] = argmax(gamma[t-k])

    END IF
END FOR

```

Figure 3.5: Pseudo-code for the Fixed-Lag Smoothing algorithm

simple filtering algorithm. It is important to notice that the algorithm estimates the most probable state at each time step and not the most probable sequence as in the Viterbi algorithm.

summary

Algorithm	Complexity	Segmentation	delay / realtime (samples)
Viterbi	$O((KM)^2T)$	optimal	offline
Filtering	$O((KM)^2T)$	sub-optimal	0
Fixed-Lag Smoothing	$O(K^2M(M+L)T)$	sub-optimal	L

Table 3.1: A summary of the algorithms used, compared in terms of time delay, complexity and accuracy

For the efficiency of the evaluation procedure detailed in the next section, all the algorithms introduced in this chapter were implemented in C++.

4

Evaluation

We have introduced a specific model for complex gesture segmentation. The model is expected to improve the segmentation method proposed in previous research based on the Segmental HMM [Caramiaux et al., 2011b]. Particularly, we expect the model to be suited to situations where important speed variations arise within gestures. In order to quantify these differences and estimate the respective performance of each algorithm, an evaluation method was designed.

Once introduced the evaluation method, we report a quantitative comparison of the proposed model with the segmental HMM. Different situations are investigated which aim at defining the suitability of the algorithms in particular contexts. As the Segmental HMM is limited to offline inference we focus on optimal decoding using the Viterbi Algorithm. Then, the issue of an implementation in realtime is investigated. Different inference algorithms are compared and discussed in order to reach a compromise between the quality of the segmentation and the delay to realtime a user could allow. Finally, a case study on segmenting the gestures of a violinist aims at identifying bowing techniques such as *Spiccato* and *Pizzicato*, offering an interesting insight of the musical applications of this study.

4.1 Evaluation method

In this section, we first introduce the material used to evaluate the model, which required the creation of a specific gesture database. Based on a function which identifies different types of errors, a general evaluation method is exposed to provide quantitative indicators about the quality of the segmentation computed by each algorithm.

4.1.1 Data collection

The first step in designing an evaluation procedure is the choice of an appropriate database for evaluation. As highlighted before, we focus in this study on gestures represented as time series. As our system is not dedicated to vision-based recognition, we must exclude all the databases that emerged in the computer vision community. Considering the specific context of music and performing arts we also exclude handwriting, fingerspelling and sign language to focus on more abstract gestures. As a consequence, it seems more appropriate for our study to look for a database from the HCI community, specifically focusing on mobile interfaces.

In [Liu et al., 2009], authors propose a gesture recognition system for mobile devices based on Dynamic Time Warping (DTW). The model is evaluated on a collection of 8 gestures,

captured using accelerometers, which were identified in a previous study conducted by Nokia Research [Kela et al., 2006] to constitute a small vocabulary of iconic gestures preferred by users. The gesture vocabulary of the database is illustrated on figure 4.1.









1	2	3	4
			
5	6	7	8
			

Figure 4.1: Gesture vocabulary of the *uWave* database. The dot denote the start and arrow the end.

The gestures of the database are independent and isolated, forbidding an evaluation on a segmentation task or for continuous gesture recognition. But the collection presents an interesting gesture: a square icon – gesture 2 of the database – which can be considered as the concatenation of 4 segments: *up*, *right*, *down* and *left* – respectively gestures 5, 3, 6 and 4 of the database. We defined a segmentation task as as representing the square gesture as four time-aligned primitive shapes.

However, given that the square gesture is a single unit, no reference segmentation is accessible and segmenting signals manually revealed very difficult. As a consequence, a new gesture database was created, inspired from this square shaped icon. We chose the vocabulary shown on figure 4.2(b). The database is constituted by four square-shaped gestures, each starting at a different corner. Gestures were recorded using a *Wii Remote*, primary controller of the video game console *Wii* by Nintendo® (figure 4.2(a)) which includes a three-axis accelerometer. Acceleration signals were resampled at a sampling frequency of 100 *Hz*. 8 participants were asked to repeat each gesture 10 times at three different speeds. In order to obtain a precise reference segmentation, participants had to synchronize their movements on an external tempo, by performing an edge of the square between two successive clicks. Three tempos were chosen: 60, 120 and 180 beats per minute (bpm). Recording the click track provides a precise reference segmentation of gestures. Because gestures are performed in a vertical plan, we keep only two directions of the accelerometer in the following tests.

4.1.2 Method

A major interest of creating a quite large database, containing 960 gestures, is to perform a statistical evaluation of the model on a segmentation task. With respect to the particular learning procedure which reduces the training phase to learning from a single example, we adopted the following general evaluation procedure. First, for each tempo, each gesture, a participant is chosen and a trial is taken as reference. Second, this trial gesture is segmented given a reference segmentation to constitute a training set of primitive gestures. In this study we consider two different reference segmentations which are discussed in the next paragraph. Finally, another trial is then taken as input signal and segmented by the model. Different situations are investigated:

- ▷ **same subject, same tempo:** reference and input signals are extracted from ges-

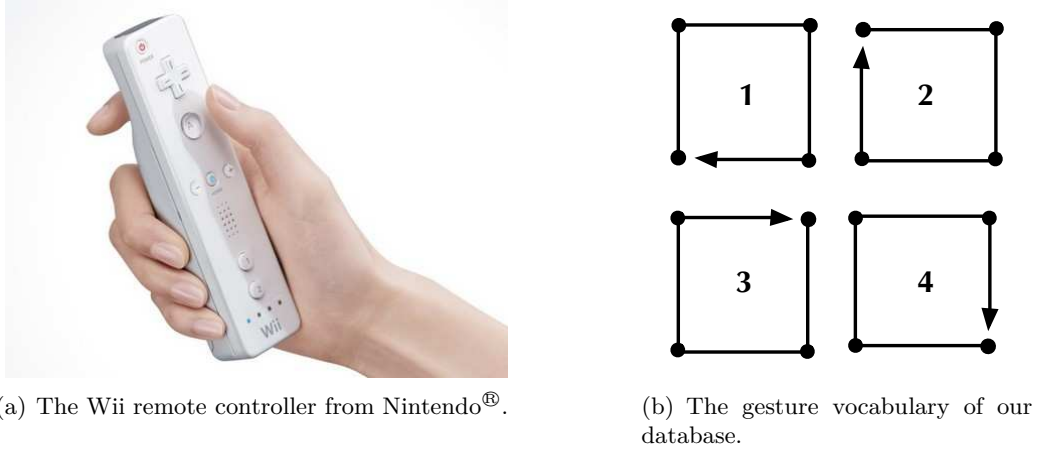


Figure 4.2: Our gesture database

tures performed by the same participant at the same speed.

- ▷ **inter-subject, same tempo:** reference and input signals are performed at the same speed by two different participants.
- ▷ **same subject, inter-tempo:** reference and input signals are performed by the same participant at different speeds.

This general procedure provides a statistical estimation of the performance of the algorithms. Indeed, for the same subject and the same tempo, if each trial is successively taken as reference and the nine others are segmented, 90 segmentations are computed per participant; tempo; gesture. As a result, for each tempo a total of 2880 gestures are segmented, giving access to an important base for evaluation. We can then estimate the efficiency of various algorithms by comparing the computed segmentations.

Parameters

In order to achieve a consistent comparison between the models, parameters have to be set so as to maximize the segmentation results. Parameters are optimized on the Hierarchical HMM by manually finding the local maximum of the recognition rate over a list of possible values of the parameters. Notably, the two models have common parameters carrying an identical meaning. These parameters are optimized on the Hierarchical HMM and are then equally set on the two models for the testing procedure. For each model, the high level structure, governing the transition probabilities between primitive gestures, is considered ergodic. That means that the initial probabilities of each primitive gesture are equal, as well as the transition probabilities between segments. Another important parameter to be set is the standard deviation of the normal distribution which defines the observation probability.

For the proposed model, two other parameters are of major interest. First, state transition probabilities of the *signal* level are determined by the vector:

$$LR_{prob} = [self, next, skip, skip_2]$$

where *self* sets the auto-transition probability, *next* define the probability of making a transition to the next state and *skip*, *skip₂* respectively define the probability of skipping

one or two states in the left-right topology. As our model considers one hidden state per sample of the reference gesture, these probabilities respectively correspond to looping on a sample – i.e. slowing down, – moving to the next sample, and skipping samples – i.e. speeding up. If the vector is limited to four values, the system can adapt to an increase of the speed of execution by a factor 3. Besides, setting the probabilities of reaching an *exit* state in the model is a primordial issue. In our model, this parameter defines the probability of finishing a primitive gesture, thus conditioning the possibility to make a transition to another primitive. In our test, setting to 0.1 and 0.75 the respective probability of reaching an *exit* state on the last two samples of the primitive shape was found optimal.

The Segmental HMM introduces a duration distribution constraining the possible time variations of segments. here, the durations are uniformly distributed on a range centered around the length of the reference segment, plus or minus 10 samples.

Reference segmentations

In the proposed model, the learning process consists in defining templates of the primitive shapes. In this study we chose to extract the primitives from a reference signal by manually annotating the time boundaries of each segment. In the following sections we refer to this definition of the primitive gestures as "*reference segmentation*" and we investigate two situations:

- ▷ **Tempo-based segmentation** : the click track recorded synchronously with the gestures of each participant is used as a reference segmentation. We define 4 segments as the acceleration signal between two successive clicks. This segmentation is directly linked to the position and corresponds to the visual representation of the gesture, as represented on the top of figure 4.3
- ▷ **Signal-based segmentation** : as highlighted by the middle plot of figure 4.3, the acceleration signals show an important activity in three areas corresponding to the corners of the squares. Focusing on the signal itself instead of considering the visual representation of the gesture leads to identify 3 primitive shapes defined by the acceleration patterns. This segmentation is defined manually by identifying the separations between two patterns. The bottom curve of figure 4.3 depicts the variance of the acceleration signals computed over the 10 trials. Interestingly, the minima of the variance – or the crossings between the variance on each axis – are a powerful indicator of this segmentation.

4.1.3 Evaluation function

In order to quantify the quality of a segmentation, an evaluation function has been defined, inspired from the evaluation method proposed for the MIREX score following contest [Cont et al., 2007]. Given a musical score, recorded as a midi file, a score following system aims at aligning the performance of musicians to the score by recognizing musical events such as notes. Thus, the systems outputs a series of symbols with a given timing and duration.

Here, the problem is fairly similar, as segmenting a gesture in primitive shapes amounts to computing the optimal series of segment with a given temporal alignment. The proposed evaluation function identifies different types of errors to quantify the quality of the segmentation, a correct segmentation requiring the recognition of the adequate sequence

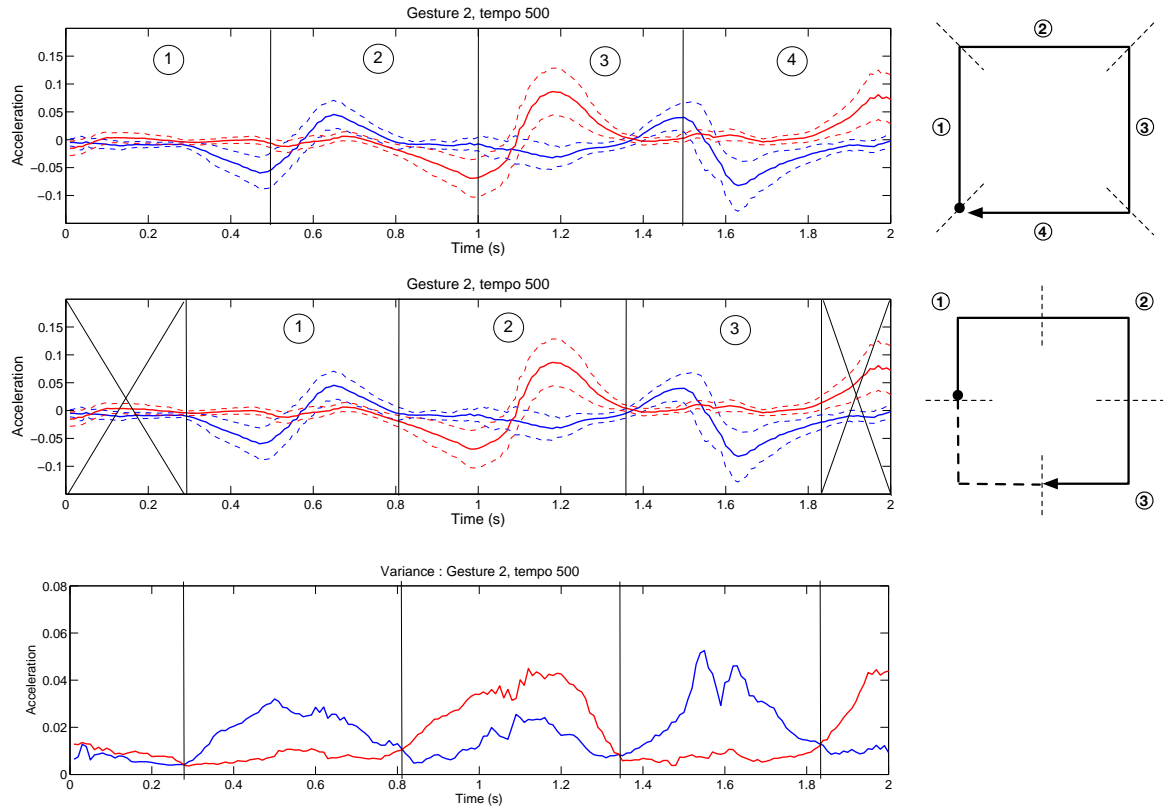


Figure 4.3: The two reference segmentations considered in this study. Mean of the acceleration signals of gesture 4 at tempo 120 *bpm* for subject 1. Dashed curves represent standard deviation of the 10 trial around the mean. The tempo-based segmentation is shown on the top curve and the bottom plot describes the signal-based segmentation.

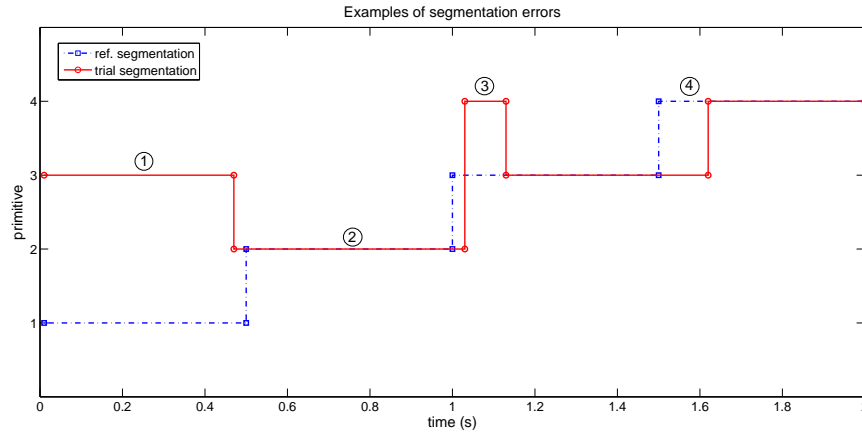


Figure 4.4: The different types of errors identified by the evaluation function: *substitution* (1), *good segment* (2), *insertion* (3) and *misaligned segment* (4).

of primitive shapes with a precise time alignment. An hypothetical segmentation result is represented on figure 4.4 which illustrates the various errors we need to recognize. The figure depicts two stair plots, the reference segmentation is represented in dashed blue line and the computed segmentation in solid red line. Each horizontal portion is a recognized segment which length defines its temporal extension. This representation provides

a visual analysis of the segmentation in comparison with the reference, both on segment identification and time alignment.

We define four types of errors. First, a *substitution* (1) is found if a wrong symbol is identified on a whole of a segment. Second, a correct segment (2) is identified if the symbol is correct on a whole segment and aligned on the reference segmentation under a threshold fixed a priori. On the figure, we observe a substitution of short duration at the beginning of the third segment (3). As it does not imply the whole segment to be wrong, this error is called an *insertion*. Finally, we define as *misaligned* segments those which symbol is correctly recognized but which are delayed over a given threshold compared to the reference segmentation (4).

In order to extract a single consistent indicator, we denote as *good segmentations* those for which all segments are recognized and aligned under a given threshold, i.e. which contain only correct segments and no insertions. In the tests of the following sections, the threshold was fixed at 200 *ms*. This value is chosen to authorize variations in the alignment between various performances of the same gesture.

4.2 Offline segmentation: a comparison with the segmental model

As illustrated by B. Caramiaux in a recent study, offline segmentation of gesture signals provide interesting prospects for musical performance analysis [Caramiaux et al., 2011b]. Intrinsically linked to time, musical gestures can show important variation in their temporal unfolding. Thus, a model designed for gesture analysis must be able to handle speed variations within gestures.

In this section, we achieve a quantitative comparison between the proposed model and the segmental HMM. Our goal is to identify the suitability of each model in various situations. In particular, we want to estimate the efficiency of the models when particular distortions appear between the reference and test signals. For that sake, we begin with a comparison on the segmentation of gestures performed by the same participant at the same speed. Then, we investigate the robustness of the algorithms to two types of distortion introduced either by inter-subject or inter-tempo performances of the same gesture. In the first case, the different execution strategies of the participants lead to distort the shapes of the acceleration patterns, in the second case it is due to a different speed of execution. For each situation, the two reference segmentations introduced in the previous section are evaluated.

4.2.1 Same subject, same tempo

Protocol and parameters

For each gesture, the participant recorded 10 trials – one after the other – synchronized on a given tempo. As a results, weak speed variations are expected considering the repeatability between successive gestures. This consideration leads to set the transition probabilities of the production level to $LR_{prob} = [0.2, 0.6, 0.1, 0.1]$, reinforcing the transitions to the next sample, so favoring an execution at the same speed. The standard deviation of the gaussian observation probability was optimized by finding the maximum recognition rate for the proposed model over a range of possible values. For the three different tempos, 180, 120 and 60 bpm, this parameter was find optimal at 0.06, 0.05 and 0.04.

Tempo-based segmentation

The results are summarized in table 4.1, which compares the accuracy of our model with that of the Segmental HMM in terms of good segmentation rate. The score is an average of the results over 4 gestures performed by 8 participants, which provides 2880 test segmentations.

As a first observation, we may notice a difference between segmentation results for the different tempos: for the proposed model, if the results are comparable for intermediate and high speed – 97.4% at 120 and 98.9% 180 bpm – they are lower for gestures performed at 60 bpm, with a recognition rate of 89.2%. This can be explained by analyzing the gesture signals shown on figure 4.5. A comparison between gestures executed at tempos 120 and 60 bpm reveal a lower amplitude and an important variability between trials for slow gestures, making acceleration patterns more difficult to identify. If we focus on the results obtained with the segmental HMM, the same decrease of the results is observed at 60 bpm. The good segmentation rate is an average of the results of 90 test for each gesture and each participant. In order to compare the results of the two models, we performed a t-test between the lists of 32 scores obtained for each tempo. For each t-test, the rejection of the null hypothesis means that the two distributions don't have equal mean. For the three tempo, the null hypothesis cannot be rejected at the 5% significance level, showing that the results obtained with the models are equivalent. This conclusion emphasizes a good repeatability between gestures performed one after the other by the same participant.

		Segmentation results (% <i>good segmentation</i>)			
		<i>tempo-based</i>		<i>signal-based</i>	
		Hierarchical HMM	Segmental HMM	Hierarchical HMM	Segmental HMM
Tempo	180 bpm	97.4	96.9	99.2	97.7
	120 bpm	98.9	99.2	100	99.8
	60 bpm	89.2	93.0	96.5	95.3

Table 4.1: Results of the segmentation on same subject and same tempo. The table reports the percentage of good segmentation, averaged over 8 participant and 4 gestures. Two reference segmentation are shown: tempo-based and signal-based segmentations.

Signal-based segmentation

The results of both models applied to the signal-based segmentation are summarized in the right part of table 4.1. For this second segmentation, results are globally higher than those obtained on the tempo-based segmentation. However, it is important to warn the reader about such a comparison. First, the signal-based segmentation includes only three primitive gestures whereas the tempo-based involves four primitives. Statistically, the number of possible errors is reduced for the signal-based segmentation. Second, the beginning and the end of the signal are cut for the signal-based segmentation, the input signals considered in each reference segmentation are then different. As a result, we cannot conclude to a higher efficiency of the models using the signal-based reference segmentation but several observations derive from these results. The better results obtained with the signal-based segmentation highlight that the selected region concentrate the most salient acceleration

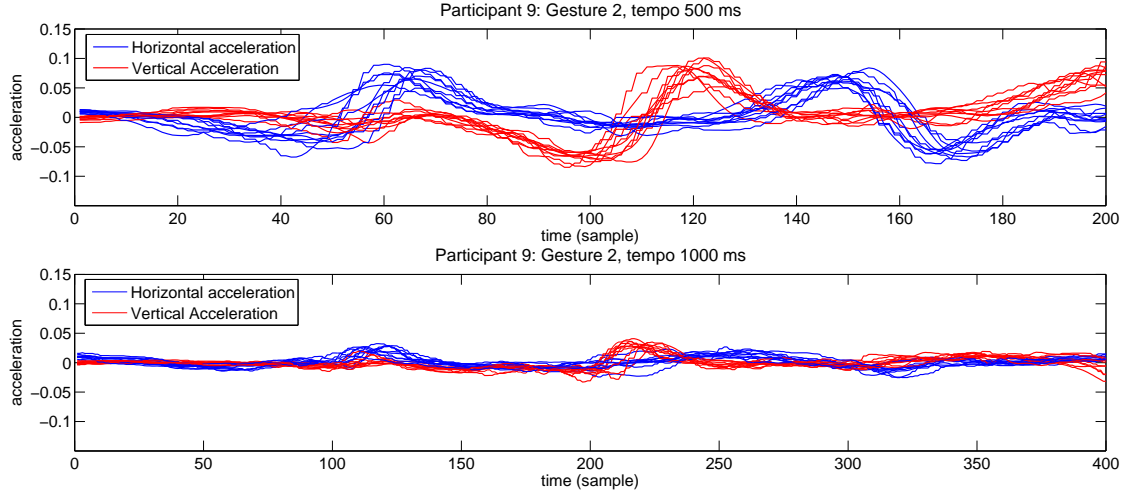


Figure 4.5: Acceleration signals of gesture 2 performed at 2 different speeds by participant 9. The 10 trials of the participant are superposed, and two axes are represented.

patterns. This reflects a greater variability at the beginning and the end of the gestures. When they perform the three corners of the square, the gestures of the participants are constrained by the goal of reaching the following corner. At the contrary, they start from a static pose and the end of the movement anticipates a relaxation, introducing variable behaviors.

Nevertheless, the comparison between the two models leads to the conclusions derived from the results on the tempo-based segmentation. At the 5% significance level, the null hypothesis of the t-test is not rejected, confirming the equivalence between the proposed model and the SHMM for gestures showing weak speed variations.

Conclusion

We have investigated same subject, same tempo segmentation. Results show that the two models are equally suitable for this situation, where weak distortions of the signals are observed due to the repeatability of the participants between various trials of the same gesture. Also, it appears that gestures performed at 60 bpm are more difficult to segment than faster gestures, because of a small amplitude and variability. In the two following sections, we evaluate the efficiency of both algorithms in situations involving important distortion.

4.2.2 Inter-subject, same tempo

Protocol and parameters

In this section, the input signal and the primitive shapes are extracted from the gestures of two distinct participants. Distortion is expected to be introduced here because different participants perform the gesture in different manners, resulting in distinct shapes of the acceleration signals. The parameters are identical to those used in the previous section.

Tempo-based segmentation

The results obtained with each algorithm are reported in the left side of table 4.2, averaged over 3200 tests. We observe that the results are lower than for *same-subject* segmentation,

which confirms the hypothesis of signal distortion. Moreover, the gap between intermediate and slow tempos is even wider: at 120 bpm about 86.4% of the gestures are well segmented against only 50.2% at 60 bpm. Different strategies can be chosen for executing the movement as shown on figure 4.6, where the mean acceleration signals of slow performances of gesture 2 are plotted for two participants. In the first case, the participant chose to move very regularly, changing direction when hearing the click. The gestures of the second participant indicates more salient acceleration patterns because it chose to hit the corners on the clicks rather than keep a constant speed. The decrease of the results at slow speed is an indicator of various strategies in the execution of the gestures when the tempo is low. As before, we performed a t-test between the results of each model to compare their respective efficiency. Here, the null hypothesis is rejected at 180 bpm at the 5% significance level, but the hypothesis cannot be rejected at 120 and 60 bpm. This means that our model is more efficient at 180 bpm, but its superiority at 120 bpm is not verified. However, the null hypothesis is rejected at 120 bpm at the 7% significance level. Thus, it appears that the proposed model is more likely to handle distortion due to various execution strategies if the gesture is performed fast.

		Segmentation results (% <i>good segmentation</i>)			
		<i>tempo-based</i>		<i>signal-based</i>	
		Hierarchical HMM	Segmental HMM	Hierarchical HMM	Segmental HMM
Tempo	180 bpm	78.9	72.9	92.3	87
	120 bpm	86.4	81.5	94.2	89.4
	60 bpm	50.2	54.8	77.1	72.1

Table 4.2: Results of the segmentation on inter-subject and same tempo. The table reports the percentage of *good segmentation*, averaged over 8 participant and 4 gestures. Two reference segmentation are shown: tempo-based and signal-based segmentations.

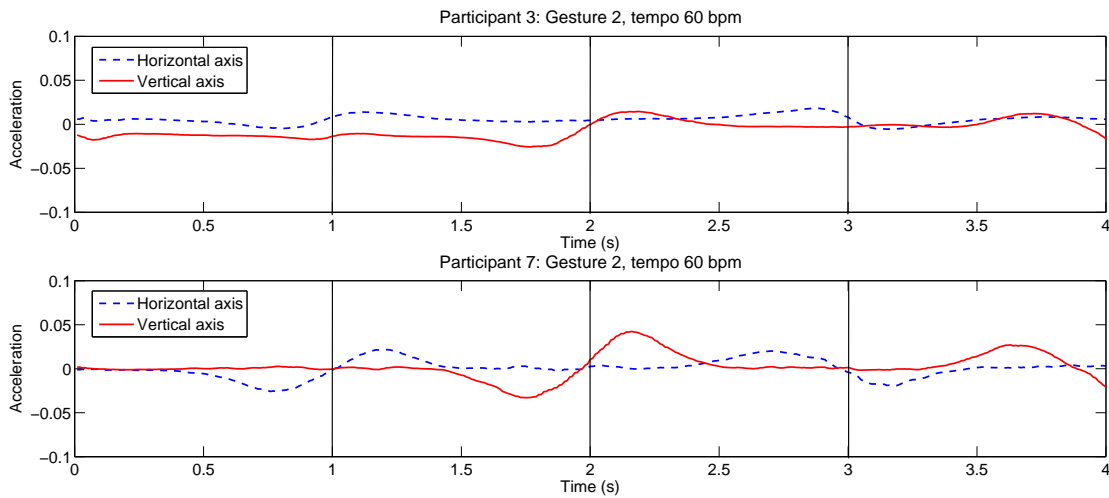


Figure 4.6: Acceleration signals of gesture 2 respectively performed by participants 3 and 7 at 60 bpm. Plain line represent the mean acceleration over 10 trials, and dashed lines represent the standard deviation around the mean curve.

Signal-based segmentation

As previously, the signal-based segmentation shows better results with both models compared to the tempo-based segmentation. Two important observations have to be made here. First, it appears that the gap between the results at high and low tempos is reduced for the HHMM as for the SHMM. As participant start from a static pose, the beginning of the gestures is often close to zero. With the tempo-based segmentation, if we consider an input gesture of small amplitude, fitting any segment to the first primitive becomes highly probable because it can be considered as a noisy signal centered around zero. Second, performing a t-test between the results of each model proves a superiority of the proposed model at 120 and 180 bpm at the 5% significance level. Moreover, if not confirmed by a t-test, the results of our model are superior to those of the segmental HMM at 60 bpm, which was false for the tempo-based segmentation. These two observations highlight that the SHMM has a better ability to handle noisy inputs of small amplitude, but that our model is able to fit the distortions of the acceleration patterns that characterize gestures executed by different participants.

Conclusion

Finally, the results show that the segmental HMM is more robust to noise, because it performs a regression on a whole segment, whereas the proposed model works at a sample level. However, as shown by the results on the signal-based segmentation, our model has a better ability to handle time variations implying a distortion of the primitive shapes. In order to confirm this conclusion, we investigate inter-tempo segmentation which implies even larger distortions between the reference and the input signal.

4.2.3 Same subject, inter-tempo segmentation**Protocol and parameters**

In this section we investigate *inter-tempo* segmentation, meaning that the reference and input signals are gestures performed at a different speed. Contrary to the same-tempo situation, we expect important variations in the speed of execution between reference and test signals. Thus, transition probabilities were adapted to allow positive or negative speed variations: $LR_{prob} = [0.25, 0.25, 0.25, 0.25]$. The standard deviation of the observation probability distribution was optimized by the procedure explained in section 4.1.2 and fixed to 0.01.

Tempo-based segmentation

The good segmentation rate computed for each couple of reference and test tempos are reported in table 4.3, averaged over 3200 tests.

Let's consider the situations for which the input gesture is performed at a higher tempo than the reference. For the proposed model, the best score is obtained between 120 and 180 bpm, namely for an increase of the speed by a factor 1.5. With 90% of good segmentation, our model outperforms the segmental HMM which shows a score of 73%. When doubling the speed, the difference is less important between the models, with 70.2% for our model against 60.2%. For that case, a t-test does not reject the null hypothesis at the 5% significance level and we cannot conclude to a superiority of the proposed model. Between 60 and 180 bpm, the Hierarchical HMM achieves 0% of good segmentation. This very low score is due to the transition probabilities of the signal level, which only authorizes to skip two states, limiting the possible speed up to a factor 3. Here, if this factor is

respected on a whole gesture, local speed variations can exceed this threshold. Setting the transition vector to $LR_{prob} = [0.2; 0.2; 0.2; 0.2; 0.2]$ allows for local speed variations of a factor 4, increasing the results from 0 to 21.2%. Situations involving an increase of the speed between the reference and the test gesture tend to highlight a better precision of our model, but this is not systematically confirmed by a t-test.

		Test Tempo (bpm)		
		180	120	60
Reference Tempo (bpm)	180		45.8	0
	120	90		3.5
	60	0	70.2	

Hierarchical HMM

		Test Tempo (bpm)		
		180	120	60
Reference Tempo (bpm)	180		24.2	0
	120	73		3.5
	60	15	60.2	

Segmental HMM

Table 4.3: *Inter-tempo* segmentation: evaluation on the tempo-based segmentation.

Considering the top right part of the table, it is evident that the results of a segmentation between a fast reference and a slow test gesture are less precise than for speeding up. Indeed, from 180 to 120 bpm the respective scores of each model are 45.8% and 24.2%, and fall to 3.5% and 0% for both models for test gestures executed at 60 bpm. here, only slowing down by a factor 1.5 reveals a more precise segmentation using the Hierarchical HMM. In previous section, we drew attention to the difficulty of segmenting slow gestures. Again, gestures performed at 60 bpm reveal very difficult to identify.

Finally, the proposed model outperforms the segmental HMM for variations of the speed of execution of a factor 1.5 or 2. In extreme cases and for segmenting very slow gestures, the model presents very low segmentation rates. Notably, tripling the speed of execution induces too much distortion to achieve an efficient segmentation. As before, segmenting slow gestures reveals a difficult task because of the noise and the small amplitude of gestures performed at 60 bpm.

Signal-based segmentation

The good segmentation rates of each model on the signal-based segmentation are reported in table 4.4. Like for the tempo-based reference segmentation, the best results are obtained for speeding up of a factor 1.5 or 2, with respective scores of 98.5% and 93.2%. For these cases, the segmentation rate of the SHMM is lower than for the first reference segmentation and our model really outperforms the Segmental HMM. As for the tempo-based segmentation, setting the transition probabilities of the production level to authorize skipping three states improves the results between 60 and 180 bpm from 21.2% to 67.2%, whereas the segmental HMM reaches 21.2%. For this reference segmentation, the proposed model show far better precision on segmenting gestures performed speedier than the reference.

For same-tempo segmentation, the signal-based reference segmentation always shows better results than that based on the tempo. Here, the Segmental HMM even show lower scores for the signal-based segmentation when speeding up and no real improvement is observed for slowing down. Indeed, between 333 and 500 bpm, the results of the signal-based segmentation are equal to 87% against 45.8% for tempo-based segmentation. Thus, we can conclude that this definition of the primitive shapes – avoiding the beginning and

the end of the gestures – is very suitable to the proposed model and confirms its weakness in presence of noise. At the contrary, the global tendency for the SHMM is a decrease of the results with signal-based segmentation, which testifies of a lack of adaptation to the deformation of the acceleration patterns.

		Test Tempo (bpm)		
		180	120	60
Reference Tempo (bpm)	180		87	3.4
	120	98.5		21.3
	60	21.2	93.2	

Hierarchical HMM

		Test Tempo (bpm)		
		180	120	60
Reference Tempo (bpm)	180		25.3	4.5
	120	47.5		10.2
	60	21.2	28	

Segmental HMM

Table 4.4: *Inter-tempo* segmentation: evaluation on the signal-based segmentation.

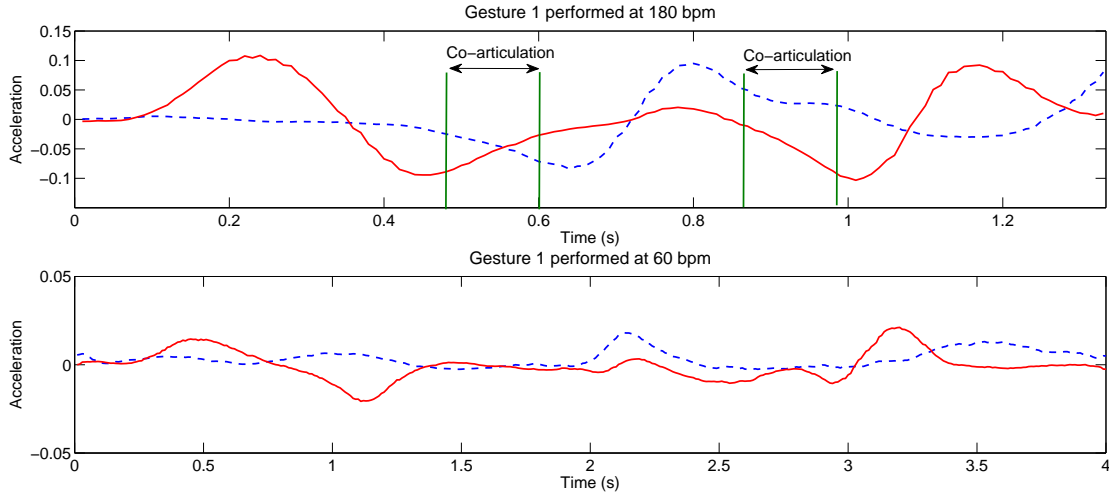


Figure 4.7: Coarticulation effects

As highlighted when studying synchronized gestures, a wide gap exist between rapid and slow gestures. For inter-tempo segmentation, this difference is even more significant: between 120 and 60 bpm the accuracy is about 21.3%, and falls to 3.4% between 180 and 60 bpm. In order to explain these results, two signals are plotted on figure 4.7: the top curve represents gesture 1 performed at 180 bpm by participant 9, and the same gesture executed at 60 bpm is represented on the bottom plot. The slow gesture puts in evidence a clear separation of the acceleration patterns, a short lapse of silence being inserted between each motif. A quite different behavior is observed at high speed: because the participant must trace the shape precisely in a short time interval, segments are sequenced quickly. This constraint is reflected in the acceleration signals by an overlapping of acceleration patterns on the top curve. This observation means that sequenced primitive gestures influence each other, introducing coarticulation. This phenomenon explains the bad results obtained for an input signal at 60 bpm and primitive shapes learned at a higher tempo.

4.2.4 Discussion

We have detailed an extensive evaluation procedure which aimed at comparing the proposed model with the Segmental HMM on a segmentation task. The models are equivalent when weak distortions of the signal are observed between the reference shapes and the input gestures. Often, segmenting slow gestures is very challenging due to the small amplitudes of the acceleration signals. Two types of distortion of the acceleration patterns have been studied, introduced by inter-subject or inter-tempo segmentation. In both cases, our models show a better accuracy, especially at high speed, which confirms its ability to handle distortion of the gesture signals due to nonlinear time variations. Thanks to the two reference segmentations introduced, we showed that the segmental HMM lacks of a dynamic fitting of the speed variations, but has an ability to smooth noisy input. A major benefit of the proposed model is the access to the time progression which enables us to study the time alignment between various executions of the same gesture.

4.3 Towards a realtime system

In the previous section, a comprehensive comparison of our model with the SHMM has been conducted, proving the ability of the proposed model to handle important speed variations. However, the method based on the Viterbi algorithm requires the entire observation sequence, forbidding a use in realtime. If this approach offers an interesting tool for studying musical performance from gesture analysis, a realtime system would open new perspectives for interactive musical systems.

In this section, we investigate several techniques to develop a realtime segmentation method. Here, the interest of expressing Hierarchical HMMs as Dynamic Bayesian Networks is even more evident. The time complexity of the algorithms derived from this framework is linear in the length of the observation sequence, ensuring to keep a constant number of operations at each time step. Moreover, different variations of the *Frontier* algorithm enables us to develop several methods for realtime or quasi-realtime gesture segmentation.

We begin by introducing a causal inference technique based on the forward algorithm. Then, an implementation of a Fixed-Lag Smoothing algorithm aims at improving the accuracy of the segmentation. A drawback of this second method is that smoothing involves a backward pass on a fixed lag, delaying the results of a few samples.

As we focus on developing a realtime system, only the tempo-based reference segmentation is considered in this section. Indeed, this reference segmentation is intuitive because related to the visual representation of the gesture so it is more suitable to a performance situation for which the signal-based segmentation would be harder to put in practice.

4.3.1 Forward algorithm

The frontier algorithm is a generalization for DBNs of the Forward-Backward algorithm. The forward algorithm is analogous that of HMMs and consists in updating the forward variable at each time step. We propose here a causal segmentation method for realtime segmentation. At each time step, the forward variable is updated, evaluating the probability of being in the macro-state X_t given the observation sequence up to the current time: $P(X_t|y_{1:t})$. Thus, a segmentation can be derived by maximizing this quantity over

all primitive shapes. The index of the most probable production state within the reference primitive allows for estimating the time progression within the gesture.

Parameters

The standard deviation was respectively set to 0.1, 0.075 and 0.025 for tempos 180, 120 and 60 bpm. As before, the probabilities of reaching an *exit* state were set to zero except for the last two samples. We focus on *same-tempo* segmentation, so that left-right transition probabilities favor a transition to the next sample and the vector is set to $LR_{prob} = [0.2 ; 0.6 ; 0.1 ; 0.1]$.

Results

In table 4.5, the results obtained with the forward algorithm are compared with an optimal decoding, for a reference segmentation based on the tempo. As we are performing causal inference, the segmentation method is sub-optimal and its accuracy is inferior to that of the Viterbi algorithm. The relationship between the segmentation rate and the speed of the gesture is verified here and is even more important for the forward algorithm, which shows a fall of the score from 64.5% to 31.4% between 120 and 60 bpm. The sensitivity to noise of our model derives from the fact it is working at a sample level, which is all the more important for the forward algorithm which provides a causal estimation of the state sequence.

		Segmentation results (% <i>good segmentation</i>)	
		Forward algorithm	Viterbi algorithm
Tempo	180 bpm	75.8	97.4
	120 bpm	64.5	98.9
	60 bpm	31.4	89.2

Table 4.5: Comparison of the Forward algorithm with the Viterbi Algorithm for same-subject and same-tempo segmentation. The results report the segmentation rate obtained on the tempo-based segmentation, averaged over 2880 test.

As the results are lower than those of the Viterbi algorithm, it is important to detail the errors introduced by the forward algorithm. The histogram of the lengths of the errors – insertions and substitutions – identified by the evaluation function is depicted on figure 4.8. As shown on the figure, the number of errors grows importantly as the tempo decreases. A more significant observation derives from the shape of the duration histogram which follows an decreasing distribution. With the Viterbi algorithm, the major part of errors are substitutions on a whole segment. At the contrary, the forward algorithm introduces a great number of insertions of short duration which don't involve the whole segment to be wrong. In fact, the causal algorithm estimates the most probable primitive at each time step, whereas the Viterbi algorithm computes the optimal sequence of primitives. Accordingly, the state sequence estimated by the forward algorithm can include forbidden transition by updating the most likely hidden state without taking into account the path up to that moment. Notably, if a portion of a primitive shape locally maximizes the probability, it will be inserted within a segment without reaching an *exit* state and make a high level transition.

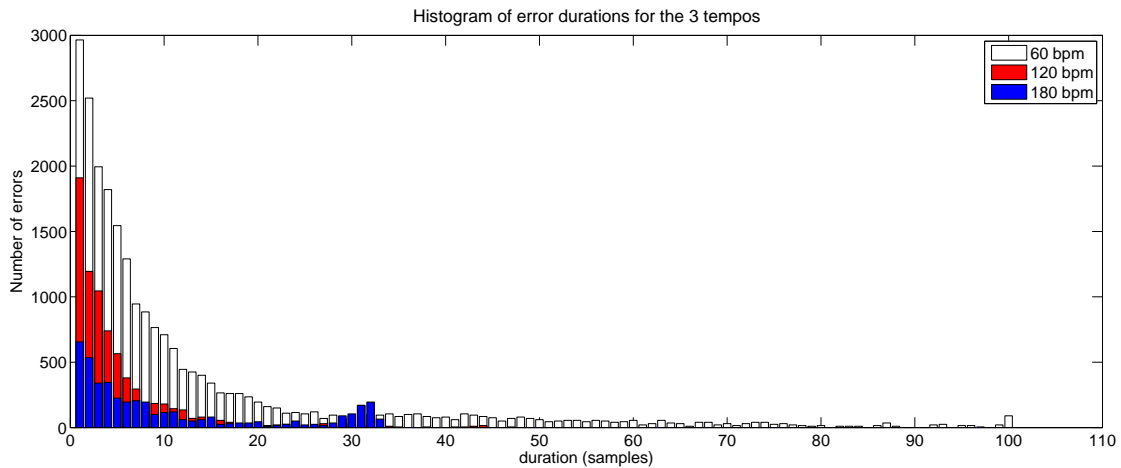


Figure 4.8: Histogram of the length of the errors (substitutions + insertions) for the Forward algorithm.

A typical example of such a phenomenon is presented on figure 4.9. The gesture performed by participant 2 at 60 bpm has a small amplitude and present low frequency oscillations. Locally, some patterns of the input signal fit exactly some portions of the reference gesture, inserting errors of short duration. The Viterbi algorithm performs an optimal decoding in the sense that it maximizes the probability of the sequence, forbidding these insertions. However, in the context of musical performance, it is evident that insertions are preferable to substitutions. The ability of the model to realign on the input signal in realtime is then an advantage in a real situation.

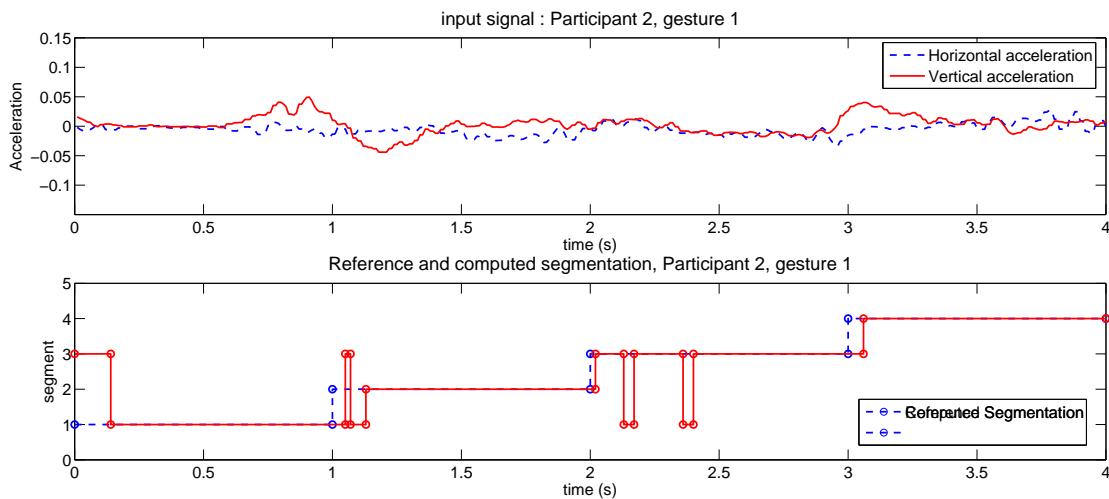


Figure 4.9: A typical example of segmentation computed with the forward algorithm presenting insertions. Executed at 60 bpm, this gesture of participant 2 has a very small amplitude and low frequency noise. As local portions of the signal fit perfectly some portions in the reference gesture, *insertions* are detected along the movement.

Our model aims at adding a hierarchy to the *Gesture Follower* developed in previous research. The lack of a high-level modeling makes this last model very poor for precise segmentation with an accuracy near to zero. The *exit* states introduced in the Hierarchical

HMM are particularly interesting as they increase the probability of the first samples of the primitive shapes when a segment is about to finish, providing a kind of automatic "reinitialization" mechanism.

4.3.2 Fixed-Lag Smoothing

As proved in the previous section, many insertions arise when segmenting gestures using the simple forward algorithm. As shown by the histogram of figure 4.8, the shorter the insertion, the numerous they are. In some situation, one could be interested in having a precise segmentation, even if that means conceding a delay to realtime. The goal would be to define a system able to smooth the forward segmentation to avoid insertions.

Again, DBNs allows for deriving such algorithms. We are particularly interested here in the Fixed-Lag Smoothing algorithm detailed in section 3.2.3, which estimates the most probable state at time $t - L$ given the observation sequence up to time t . The system then continuously updates the same parameters – primitive and time progression – at each time step, but with a delay equal to the lag. We detail here some segmentation results obtained with this method in order to find a compromise between the quality of the estimated segmentation and the delay a user could allow.

Figure 4.10 depicts the results obtained for segmenting gestures given the tempo-based reference segmentation. The good segmentation rate is evaluated as a function of the lag and compared with the results of the forward algorithm. To achieve a consistent comparison, the same parameters are set for each algorithm, the standard deviation being optimized on the forward algorithm.

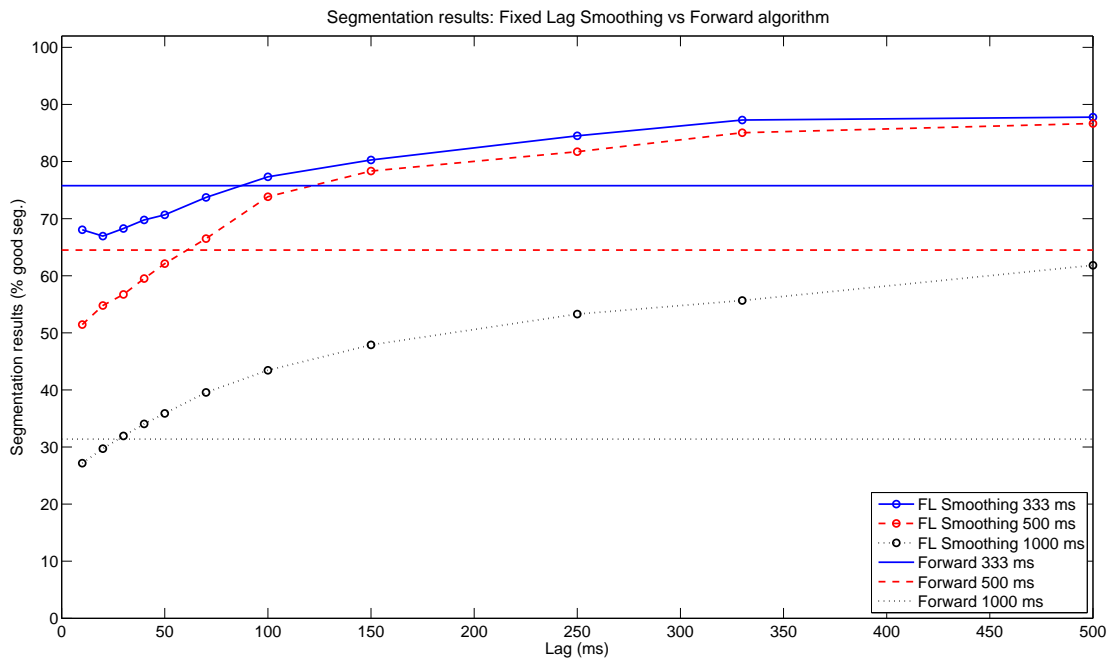


Figure 4.10: Fixed-Lag Smoothing: Influence of the lag and comparison with the Forward algorithm

As a first observation, we may notice that the accuracy increases with the lag, which confirms the smoothing effect expected: the more the lag is large, the more the backward

pass is able to smooth and avoid insertions. However, we may notice that for short lags, the algorithm performs lower than the simple forward pass. We observe that the forward algorithm is very efficient for short gestures, performed at 180 bpm. For this tempo, a delay of almost 100 ms is needed for the smoothing algorithm to show better accuracy. At the contrary, the forward algorithm is very sensitive to insertions at 60 bpm and the smoothing algorithm overtakes the forward pass from 30 ms, namely for a lag of 3 samples.

4.3.3 Discussion

Globally, the realtime algorithms presented in this section are less precise than the optimal decoding using the Viterbi algorithm. Notably, a great number of insertions arise when using the Forward algorithm, the length of the errors following a decreasing distribution. The number of insertions increases as the speed decreases. Fixed-Lag smoothing has been studied as a compromise between a smoothing effects which avoids insertions, and a delay to realtime. The accuracy of the algorithm increases with the lag, confirming the smoothing effect. For each tempo, Fixed-Lag smoothing outperforms the Forward algorithm after a given lag, and this threshold grows as the speed increases.

Finally, it would be interesting to implement an adaptive system taking advantage of the performance of the forward algorithm at high speed and of the smoothing for slow gestures. Defining the lag dynamically as a function of the estimated speed would optimize the results of realtime segmentation. Moreover, this perspective is coherent in the sense that rapid gestures require short lags to remain reactive, but slow gestures can accept a longer delay.

4.4 A musical application: segmentation of violin bowing techniques

If the previous method was designed to evaluate the suitability of our model for gesture analysis, the database used can seem far from what we could consider as "musical" or "expressive" gestures. We propose in this section an application to segmenting the movements of a violinist. In the IMTR team, two types of research have been conducted: analyzing the gestures of violinists and designing a system which involved developing both specific sensors and realtime gesture analysis software.

Notably, a collaboration with the french composer Florence Baschet raised interest in studying bowing techniques. Captured using miniature sensors fixed on the bow, the acceleration was analyzed in order to correlate the gestures of the instrumentalist with bowing techniques such as *Spiccato* and *Pizzicato*. Embedded in a realtime system, the gesture analysis technique is central to Florence Baschet's quartet *StreicherKreis*¹, created at Ircam in 2007.

In the case study presented here, our model is used for segmenting violin bowing techniques. We consider two recordings of the same musical phrase, which include an audio track and the signals from a three-axis accelerometer. On figure 4.11, the musical score is correlated with the audio track and the acceleration signal in the direction of the bow. Different bow strokes appear in the score, such as *Spiccato*, *Col Legno* and *Pizzicato*. Two trials of the same phrase are presented on the figure, performed by the same player on the same day. As reference segmentation we consider the intrinsic divisions of the musical score, represented by dashed lines which emphasizes the temporal alignment between the

¹<http://www.florencebaschet.com/site/fiche9-StreicherKreis.html>

scores and the two performances. The segmentation task was set as follows: the first performance is segmented manually, given the audio track and the musical score. The twelve primitive shapes obtained are learned and the accelerations signals of the second performance are segmented by the model.

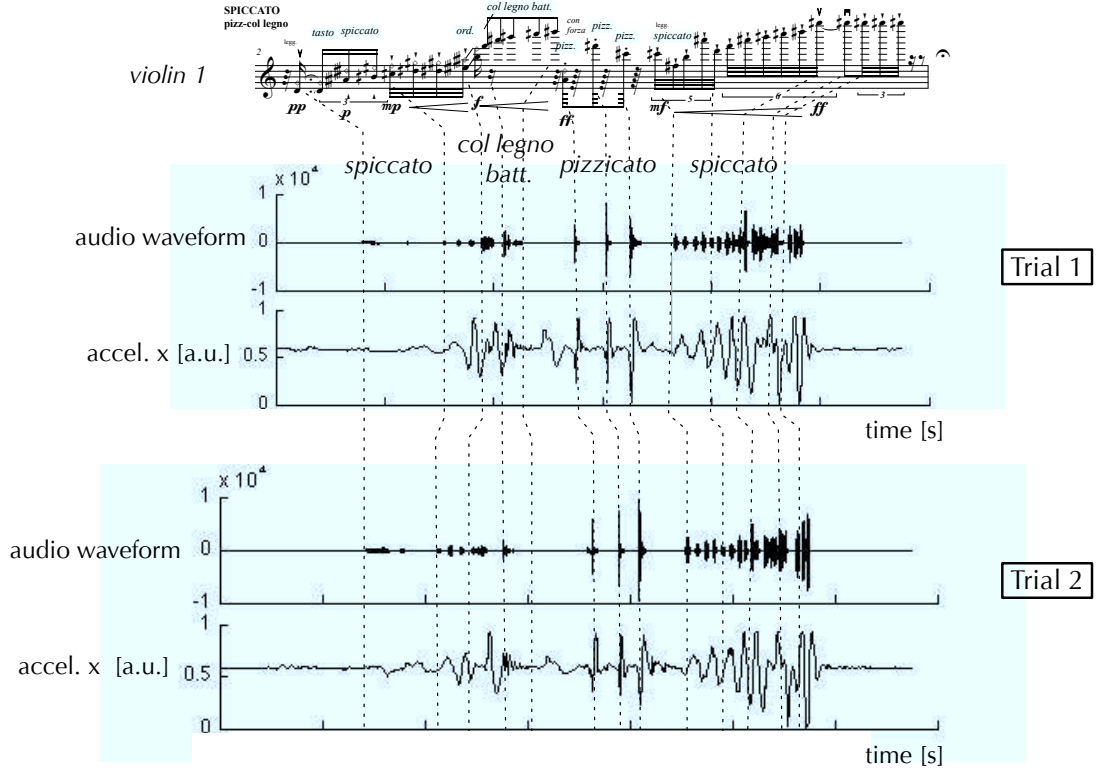


Figure 4.11: The musical score of the violin phrase, correlated with the audio track and an acceleration signal. The phrase includes different bowing techniques: *Spiccato*, *Col Legno* and *Pizzicato*. The time alignment between the score, the audio track and the acceleration signal is represented by vertical dashed lines. Two trials are presented, performed by the same player on the same day.

Resulting segmentations are plotted on figure 4.12. First, the Viterbi algorithm provides an almost perfect segmentation of the gesture, as shown on the second curve of figure 4.12. In fact, only two substitutions are introduced. Segment 9 is detected in place of segment 2, but they correspond to the same bow stroke, namely *Spiccato*. Similarly, the sixth segment is substituted to the seventh, making a confusion between two impulsions of *Pizzicato*. Thus, the errors introduced are not aberrant as they correspond to the same bowing technique.

The same type of error is introduced by the Forward algorithm between segments 6 and 7, as shown on the third curve. However, the causal segmentation presents a lot of insertions, by a majority situated at the transitions between segments. These insertions have a length inferior to a few samples. So, in a concert situation for example, errors will only be introduced on a duration inferior to 30 ms at the transitions between segments.

In order to improve the precision of the realtime segmentation, we applied Fixed-Lag smoothing to the same segmentation task. The segmentation computed with a lag of 25 samples is plotted on the bottom curve of the figure. Only a few insertions remain and

the segmentation is close to the results of the Viterbi algorithm. However, these results require a lag of 25 samples, inducing a delay of 110 ms. Depending on the needs of a user, one could chose either to accept insertions of 30 ms or concede a delay to smooth the segmentation results.

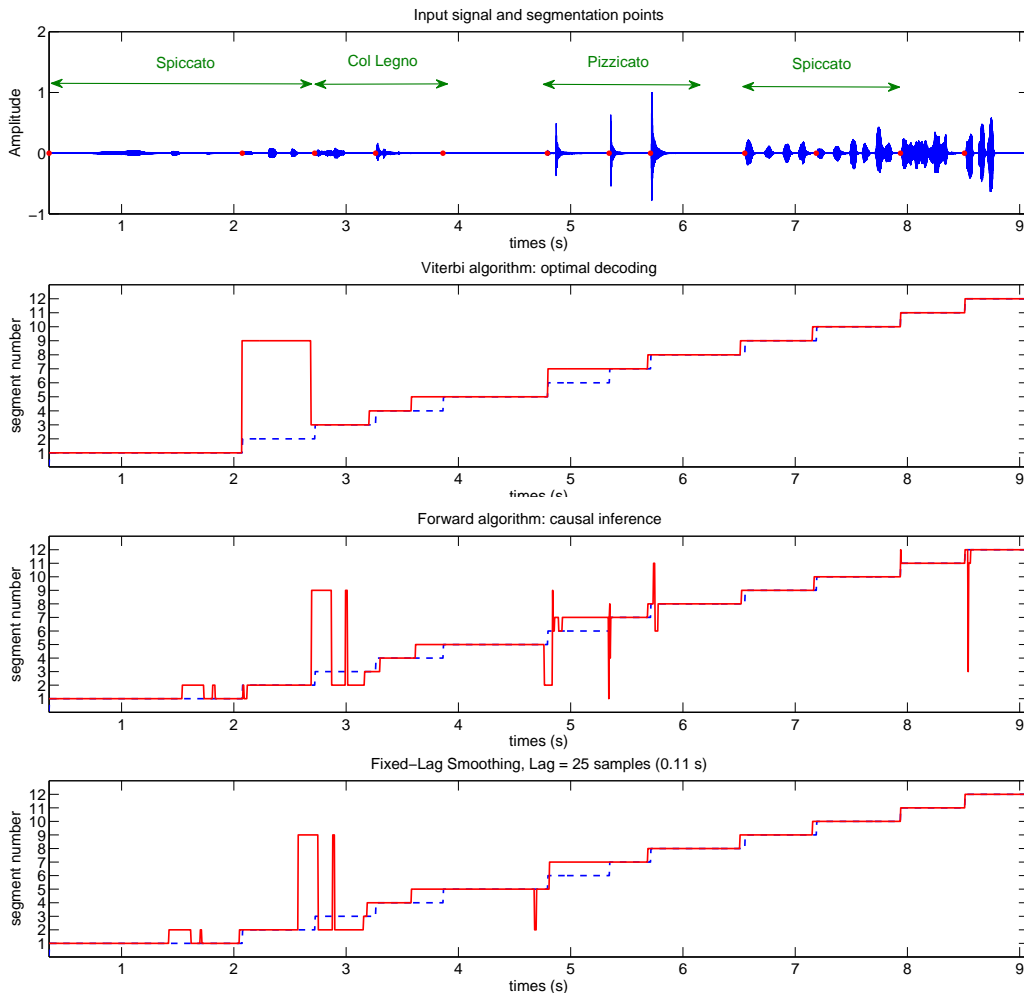


Figure 4.12: Segmentation results on the violin phrase compared for three algorithms. On each figure, the top curve reports the audio waveform and the reference segmentation points. On the second curve, the segmentation computed with the Viterbi algorithm (red) is compared with the reference segmentation (dashed blue line). The third curve reports the results obtained with the forward algorithm, and the bottom plot reports the results of Fixed-Lag Smoothing with a lag of 25 samples, corresponding to a delay of 110 ms

5

Conclusion and future directions

We have proposed a model for realtime segmentation of complex gestures. Based on the Hierarchical Hidden Markov Model, the system allows for representing gestures as a sequence of primitive shapes. A particular topology provides a time precision at the sample level and enables learning templates from a single example. Efficient algorithms derive from the formalization as a Dynamic Bayesian Network: the Viterbi algorithm performs offline decoding and two realtime methods provide approximate solutions.

The evaluation of the offline algorithm on an accelerometer database provided a quantitative comparison between our model and the segmental HMM on a segmentation task. Two types of distortion of the acceleration patterns have been studied, introduced by inter-subject or inter-tempo segmentation. In both cases, our models shows a better accuracy, especially at high speed, which confirms its ability to handle distortion of the gesture signals due to nonlinear time variations. At the contrary, the rigid temporal modeling of the segmental HMM limits its fit to distortion, but the model is able to smooth noisy input.

An evaluation on the same database compared two methods for realtime segmentation. The forward algorithm is an efficient causal segmentation technique but inserts many false detections of short durations while segmenting slow gestures. Fixed-Lag smoothing is a compromise between a smoothing effect which avoids insertions and a delay to realtime. The threshold on the lag after which the algorithm outperforms the forward procedure decreases with the speed. As a result, it would be interesting to define a system in which the lag is adapted dynamically as a function of the estimated speed of the gestures.

Finally, the example of realtime segmentation of bowing techniques illustrates a musical application of our model. Both the forward algorithm and Fixed-Lag smoothing show interesting results and provide two different strategies for a context of musical performance.

A major prospect would be to apply the system to two types of studies: performance analysis, for example by segmenting the gestures of an instrumentalist or dancer; and realtime interaction. During this study, we have developed a prototype of an external object for Max/MSP. Due to a lack of time, the object could not be optimized to reduce the computation time and memory. For gestural control of sound synthesis, such a system is interesting because the definition of a vocabulary of primitive gestures permits a realtime segmentation of complex or continuous gestures, allowing multi-level mapping strategies.

Here, we consider the high level structure as ergodic. Implementing a learning procedure of the transition probabilities between primitive shapes would improve the segmentation results, in particular avoiding insertions in realtime algorithms. We can imagine a learning process based on an extensive database of gesture sequences, but a more interesting process would define the transition probabilities dynamically given a musical score to achieve

efficient gesture "following".

The PhD thesis I begin next year aims at defining the coupling between gesture and sound in various contexts by machine learning techniques. Notably, using active learning would allow for an adaptive learning of both the primitive gestures and their relationship to music.

Finally, the evaluation has been limited to a simple database. If continuous gesture recognition and spotting is an active issue in the computer vision community, few solutions exist for mobile devices. Notably, we did not find a collection including continuous gesture sequences. A short term perspective is the creation of a new database of continuous gestures related to musical expression. Inspired from conductor gestures, the specifications of a new database are currently investigated to define complex gestures composed by a *pre-gesture* or attack, followed by a series of unit segments.

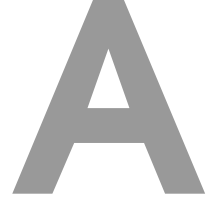
Bibliography

- [Artieres et al., 2007] Artieres, T., Marukatat, S., and Gallinari, P. (2007). Online handwritten shape recognition using segmental hidden markov models. *IEEE Trans. Pattern Anal. Mach. Intell.*, 29:205–217.
- [Baum and Petrie, 1966] Baum, L. and Petrie, T. (1966). Statistical inference for probabilistic functions of finite state markov chains. *The Annals of Mathematical Statistics*, 37(6):1554–1563.
- [Bevilacqua and Flety, 2004] Bevilacqua, F. and Flety, E. (2004). Captation et analyse du mouvement pour l’interaction entre danse et musique. In *Rencontres Musicales Pluridisciplinaires - le corps & la musique*, Lyon.
- [Bevilacqua et al., 2010] Bevilacqua, F., Zamborlin, B., Sypniewski, A., Schnell, N., Gu  dy, F., and Rasamimanana, N. (2010). Continuous realtime gesture following and recognition. In Kopp, S. and Wachsmuth, I., editors, *Gesture in Embodied Communication and Human-Computer Interaction*, volume 5934 of *Lecture Notes in Computer Science*, pages 73–84. Springer Berlin / Heidelberg.
- [Bhuyan et al., 2006] Bhuyan, M., Ghosh, D., and Bora, P. (2006). Continuous hand gesture segmentation and co-articulation detection. *Computer Vision, Graphics and Image Processing*, pages 564–575.
- [Bilmes, 2006] Bilmes, J. A. (2006). What hmms can do. *IEICE - Trans. Inf. Syst.*, E89-D:869–891.
- [Bloit et al., 2010] Bloit, J., Rasamimanana, N., and Bevilacqua, F. (2010). Modeling and segmentation of audio descriptor profiles with segmental models. *Pattern Recognition Letters*, 31(12).
- [Cadoz and Wanderley, 2000] Cadoz, C. and Wanderley, M. (2000). Gesture-music. In *Trends in Gestural control of music*. Ircam - Centre Pompidou.
- [Caramiaux et al., 2011a] Caramiaux, B., Susini, P., Bianco, T., Bevilacqua, F., Houix, O., Schnell, N., and Misdariis, N. (2011a). Gestural embodiment of environmental sounds: an experimental study. In Jensenius, A. R., Tveit, A., God  y, R. I., and Overholt, D., editors, *Proceedings of the International Conference on New Interfaces for Musical Expression*, pages 144–148, Oslo, Norway.
- [Caramiaux et al., 2011b] Caramiaux, B., Wanderley, M., and Bevilacqua, F. (2011b). Segmenting and parsing instrumentalist’s gestures. *Journal of New Music Research* (submitted).
- [Cont et al., 2007] Cont, A., Schwarz, D., Schnell, N., and Raphael, C. (2007). Evaluation of real-time audio-to-score alignment. *International Symposium on Music Information Retrieval (ISMIR)*.
- [Corradini, 2001] Corradini, A. (2001). Dynamic Time Warping for off-line recognition of a small gesture vocabulary. *Recognition, Analysis and Tracking of Faces and Gestures in Real-Time Systems, IEEE ICCV Workshop on*.

- [de Laubier and Goudard, 2006] de Laubier, S. and Goudard, V. (2006). Meta-instrument 3: a look over 17 years of practice. In *Proceedings of the 2006 conference on New interfaces for musical expression*, NIME '06, pages 288–291, Paris, France, France. IRCAM - Centre Pompidou.
- [Doucet et al., 2000] Doucet, A., Freitas, N. d., Murphy, K. P., and Russell, S. J. (2000). Rao-blackwellised particle filtering for dynamic bayesian networks. In *Proceedings of the 16th Conference on Uncertainty in Artificial Intelligence*, UAI '00, pages 176–183, San Francisco, CA, USA. Morgan Kaufmann Publishers Inc.
- [Eickeler et al., 1998] Eickeler, S., Eickeler, S., Kosmala, A., and Rigoll, G. (1998). Hidden Markov Model based continuous online gesture recognition. *International Conference on Pattern Recognition (ICPR)*, 2:1206–1208.
- [Fine et al., 1998] Fine, S., Singer, Y., and Tishby, N. (1998). The hierarchical hidden markov model: Analysis and applications. *Machine Learning*, 32:41–62.
- [Godøy, 2006] Godøy, R. I. (2006). Gestural-sonorous objects: embodied extensions of schaeffer’s conceptual apparatus. *Org. Sound*, 11:149–157.
- [Godøy et al., 2006] Godøy, R. I., Haga, E., and Jensenius, A. R. (2006). Exploring music-related gestures by sound tracing: a preliminary study. In *Proceedings of the COST287-ConGAS 2nd International Symposium on Gesture Interfaces for Multimedia Systems*.
- [Godoy et al., 2010] Godoy, R. I., Jensenius, A. R., and Nymoen, K. (July/August 2010). Chunking in music by coarticulation. *Acta Acustica united with Acustica*, 96:690–700(11).
- [Godøy and Leman, 2009] Godøy, R. I. and Leman, M. (2009). *Musical gestures: sound, movement, and meaning*. Routledge.
- [Jensenius, 2007] Jensenius, A. R. (2007). *Action-sound: Developing methods and tools to study music-related body movement*. PhD thesis, Department of Musicology, University of Oslo.
- [Kela et al., 2006] Kela, J., Korpipaa, P., Mantyjarvi, J., Kallio, S., Savino, G., Jozzo, L., and Marca, S. (2006). Accelerometer-based gesture control for a design environment. *Personal and Ubiquitous Computing*, 10(5):285–299.
- [Kendon, 2004] Kendon, A. (2004). *Gesture: Visible action as utterance*. Cambridge Univ Press.
- [Lee and Kim, 1999] Lee, H.-K. and Kim, J. H. (1999). An hmm-based threshold model approach for gesture recognition. *Pattern Analysis and Machinery Intelligence, Transactions on*.
- [Leman, 2006] Leman, M. (2006). *Embodied Music Cognition and Mediation Technology*. MIT Press.
- [Liu et al., 2009] Liu, J., Wang, Z., Zhong, L., Wickramasuriya, J., and Vasudevan, V. (2009). uwave: Accelerometer-based personalized gesture recognition and its applications. *Pervasive Computing and Communications, IEEE International Conference on*, pages 1–9.

- [Loehr and Palmer, 2007] Loehr, J. D. and Palmer, C. (2007). Cognitive and biomechanical influences in pianists' finger tapping. *Experimental Brain Research*, 178(4):518–528.
- [Malloch and Wanderley, 2007] Malloch, J. and Wanderley, M. M. (2007). The t-stick: from musical interface to musical instrument. In *Proceedings of the 7th international conference on New interfaces for musical expression*, NIME '07, pages 66–70, New York, NY, USA. ACM.
- [Mathews, 1989] Mathews, M. V. (1989). *The conductor program and mechanical baton*, pages 263–281. MIT Press, Cambridge, MA, USA.
- [Miller, 1956] Miller, G. A. (1956). The magical number seven, plus or minus two: some limits on our capacity for processing information. *Psychological Review*, 101(2):343–52.
- [Mitra and Acharya, 2007] Mitra, S. and Acharya, T. (2007). Gesture recognition: A survey. *Systems, Man, and Cybernetics, Part C: Applications and Reviews, IEEE Transactions on*, 37(3):311–324.
- [Murphy, 2002] Murphy, K. P. (2002). *Dynamic bayesian networks: representation, inference and learning*. PhD thesis, University of California, Berkeley.
- [Ostendorf et al., 1996] Ostendorf, M., Digalakis, V., and Kimball, O. A. (1996). From hmms to segment models: a unified view of stochastic modeling for speech recognition. In *Speech and Audio Processing, IEEE Transactions on*, volume 4.
- [Rabiner, 1989] Rabiner, L. (1989). A tutorial on hidden markov models and selected applications in speech recognition. *Proceedings of the IEEE*, 77(2):257 – 286.
- [Rajko et al., 2007] Rajko, S., Qian, G., Ingalls, T., and James, J. (2007). Real-time gesture recognition with minimal training requirements and on-line learning. *Computer Vision and Pattern Recognition, IEEE Computer Society Conference on*, 0:1–8.
- [Rasamimanana et al., 2011] Rasamimanana, N., Bevilacqua, F., Schnell, N., Guedy, F., Flety, E., Maestracci, C., Zamborlin, B., Frechin, J.-L., and Petrevski, U. (2011). Modular musical objects towards embodied control of digital music. In *Proceedings of the fifth international conference on Tangible, embedded, and embodied interaction*, TEI '11, pages 9–12, New York, NY, USA. ACM.
- [Rasamimanana et al., 2009] Rasamimanana, N., Kaiser, F., and Bevilacqua, F. (2009). Perspectives on gesture-sound relationships informed from acoustic instrument studies. *Organised Sound*, 14(2).
- [Rosenbaum et al., 1983] Rosenbaum, D. A., Kenny, S. B., and Derr, M. A. (1983). Hierarchical control of rapid movement sequences. *Journal of Experimental Psychology: Human Perception and Performance*, 9(1):86 – 102.
- [Schaeffer, 1966] Schaeffer, P. (1966). *Traité des Objets Musicaux*. Editions du Seuil.
- [Subramanya et al., 2007] Subramanya, A., Subramanya, A., Raj, A., and Bilmes, J. A. (2007). Hierarchical models for activity recognition. *IEEE Conference in Multimedia Processing*, pages 233–237.
- [Turaga et al., 2008] Turaga, P., Chellapa, R., Subrahmanian, V., and Udrea, O. (2008). Machine recognition of human activities: A survey. In *Circuits and Systems for Video Technology, IEEE Transactions on*, volume 18.

- [Wainwright and Jordan, 2007] Wainwright, M. and Jordan, M. (2007). Graphical models, exponential families, and variational inference. *Foundations and Trends in Machine Learning*, 1(1–2):1–305.
- [Wanderley and Battier, 2000] Wanderley, M. and Battier, M. (2000). *Trends in Gestural control of music*. Ircam - Centre Pompidou.
- [Wanderley et al., 2005] Wanderley, M., Vines, B. W., Middleton, N., McKay, C., and Hatch, W. (2005). The musical significance of clarinetists’ ancillary gestures: An exploration of the field. *Journal of New Music Research*, 34(1):97—113.
- [Widmer et al., 2003] Widmer, G., Dixon, S., Goebel, W., Pampalk, E., and Tobudic, A. (2003). Horowitz factor. *AI Magazine*, pages 111–130.
- [Wilson and Bobick, 1999a] Wilson, A. D. and Bobick, A. F. (1999a). Parametric hidden markov models for gesture recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 21(9):884–900.
- [Wilson and Bobick, 1999b] Wilson, A. D. and Bobick, A. F. (1999b). Real-time online adaptive gesture recognition. In *Proceedings of the International Workshop on Recognition, Analysis, and Tracking of Faces and Gestures in Real-Time Systems*, RATFG-RTS ’99, pages 111–, Washington, DC, USA. IEEE Computer Society.
- [Yamato et al., 1992] Yamato, J., Ohya, J., and Ishii, K. (1992). Recognizing human action in time-sequential images using hidden markov model. In *Computer Vision and Pattern Recognition, 1992. Proceedings CVPR’92., 1992 IEEE Computer Society Conference on*, pages 379–385. IEEE.
- [Zweig, 1996] Zweig, G. (1996). A forward-backward algorithm for inference in bayesian networks and an empirical comparison with HMMs. Master’s thesis, UC Berkeley CS Dept.



Model description: representation and algorithms

A.1 Representation

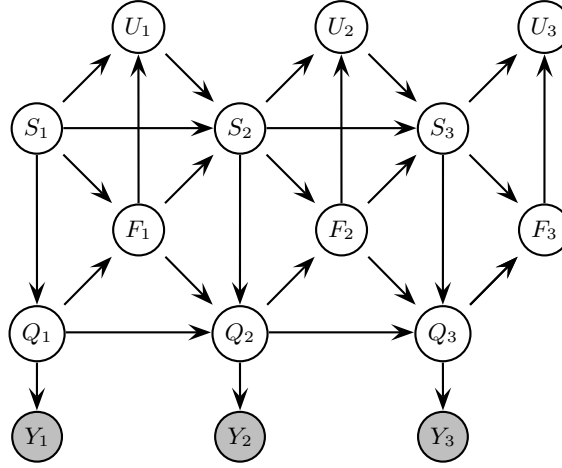


Figure A.1: The 2-level HHMM of our study, represented as a DBN. Q_t is the production state at time t , S_t is the symbolic state at time t ; $F_t = 1$ if the sub-HMM has finished (entered its exit state), otherwise $F_t = 0$. Shaded nodes are observed, the remaining nodes are hidden.

A.1.1 Notations

- ▷ $Y = y_1, y_2 \dots y_T$: Observation sequence of length T
- ▷ Q_t : Production state at time t
- ▷ S_t : Symbolic state at time t
- ▷ F_t : binary indicator that is "on" if the sub-HMM has just "finished" (is about to enter his end state)
- ▷ U_t : binary indicator that is "on" if the symbolic state sequence has just finished.
- ▷ M : number of primitive gestures = number of symbolic states
- ▷ $M^{(i)}$: Number of states of the production level called by i (length of primitive i)
- ▷ $H = \{h_i\}$: Prior probabilities for the symbolic level

$$h_i = P(S_1 = i)$$

▷ $G = (g_{il})$: State transition probability matrix for the symbolic level

$$g_{il} = P(S_{t+1} = l | S_t = i)$$

▷ $\Pi^{(i)} = \{\pi_j^{(i)}\}$: Prior probability distribution for primitive i (vertical Transition Probability)

$$\pi_j^{(i)} = P(Q_t = j | S_t = i)$$

▷ $A^{(i)} = (a_{jk}^{(i)})$: State transition probability for primitive i

$$a_{jk}^{(i)} = P(Q_{t+1} = k | Q_t = j, S_t = i)$$

▷ $B^{(i)} = (b_j^{(i)}(y_t))$: Emission probability

$$b_j^{(i)}(y_t) = P(Y_t | Q_t = j, S_t = i)$$

A.1.2 Conditional Probability Distributions

Initial probabilities : $t = 1$

$$\begin{aligned} P(S_1 = i) &= h_i \\ P(Q_1 = j | S_1 = i) &= \pi_j^{(i)} \\ U_1 &= 0 \\ F_1 &= 0 \end{aligned}$$

Production Level : $t = 2 \cdots T - 1$

$$P(Q_t = k | Q_{t-1} = j, F_{t-1} = f, S_t = i) = \begin{cases} \tilde{a}_{jk}^{(i)} & \text{if } f = 0 \\ \pi_k^{(i)} & \text{if } f = 1 \end{cases}$$

$$P(Q_t = k | Q_{t-1} = j, F_{t-1} = f, S_t = i) = (1 - f) \cdot \tilde{a}_{jk}^{(i)} + f \cdot \pi_k^{(i)}$$

where we assume $j, k \neq \text{end}$, and where $\tilde{A}^{(i)}$ is a scaled version of $A^{(i)}$ where :

$$\tilde{a}_{jk}^{(i)} = \frac{a_{jk}^{(i)}}{1 - a_{j \text{ end}}^{(i)}}$$

The equation for binary indicator F_t is :

$$P(F_t = 1 | Q_t = j, S_t = i) = a_{j \text{ end}}^{(i)}$$

$$P(F_t = f | Q_t = j, S_t = i) = f \cdot a_{j \text{ end}}^{(i)} + (1 - f)(1 - a_{j \text{ end}}^{(i)})$$

Symbolic Level : $t = 2 \cdots T - 1$

$$P(S_t = l | S_{t-1} = i, F_{t-1} = f, U_{t-1} = u) = \begin{cases} \delta(i, l) & \text{if } f = 0 \\ \tilde{g}_{il} & \text{if } f = 1 \text{ and } u = 0 \\ h_l & \text{if } f = 1 \text{ and } u = 1 \end{cases}$$

$$P(S_t = l | S_{t-1} = i, F_{t-1} = f, U_{t-1} = u) = f [u \cdot h_l + (1 - u) \tilde{g}_{il}] + (1 - f) \delta(i, l)$$

The equation for binary indicator U_t is :

$$P(U_t = 1 | S_t = i, F_t = f) = \begin{cases} 0 & \text{if } f = 0 \\ g_{i \text{ end}} & \text{if } f = 1 \end{cases}$$

$$P(U_t = u | S_t = i, F_t = f) = u f \cdot g_{i \text{ end}} + (1 - u)(1 - f \cdot g_{i \text{ end}})$$

Final Slice : $t = T$

To force all segmentations to be consistent with the length of the sequence, we must ensure that all sub-hmms have reached their final state, assuming $U_T = 1$, and $F_T = 1$.

A.2 Forward Algorithm

A.2.1 Forward pass: formalization using the frontier algorithm

In this section we apply the general *Frontier* algorithm detailed in [Murphy, 2002]. The notations are adapted to our case and respect the conventions of the previous section. The algorithm is based on the definition of a *frontier* containing nodes. The frontier is initialized at time step t by containing all nodes of the time slice. To update the frontier, nodes of the following time slice are added and nodes of time slice t are removed, until the frontier only contains the node of time slice $t + 1$. A node can be added if all its parents are in the frontier, and we can remove a node if all its children are already in the frontier.

First we need to define a variable α as the probability of being in the macro state $X_t = \{S_t, Q_t, F_t, U_t\}$:

$$\alpha_t(j, i, f, u) = P(Q_t = j, S_t = i, F_t = f, U_t = u \mid y_{1:t})$$

Let $F_{t,0} = \alpha_{t-1}(j, i, f, u)$, and consider the frontier containing all nodes in slice $t - 1$. Since all its parents are already in the frontier, we can add node S_t to the frontier :

$$\begin{aligned} F_{t,1}(j, l, i, f, u) &= P(S_t = l, S_{t-1} = i, Q_{t-1} = j, F_{t-1} = f, U_{t-1} = u \mid y_{1:t-1}) \\ &= P(S_t = l \mid S_{t-1} = i, F_{t-1} = f, U_{t-1} = u) \cdot F_{t,0}(j, i, f, u) \end{aligned}$$

As all nodes that depend on S_{t-1} and U_{t-1} are in the frontier, we can marginalize over these variables :

$$F_{t,2}(j, l, f) = \sum_i \sum_u F_{t,1}(j, l, i, f, u)$$

Add now Q_t :

$$\begin{aligned} F_{t,3}(k, j, l, f) &= P(Q_t = k, S_t = l, Q_{t-1} = j, F_{t-1} = f | y_{1:t-1}) \\ &= P(Q_t = k | Q_{t-1} = j, F_t = f, S_t = l) \cdot F_{t,2}(j, l, f) \end{aligned}$$

then remove Q_{t-1} and F_{t-1} :

$$\begin{aligned} F_{t,4}(k, l) &= P(Q_t = k, S_t = l | y_{1:t-1}) \\ &= \sum_j \sum_f F_{t,3}(k, j, l, f) \end{aligned}$$

In the same way we can add nodes F_t and U_t :

$$\begin{aligned} F_{t,5}(k, l, f) &= P(F_t = f | Q_t = k, S_t = l) \cdot F_{t,4}(k, l) \\ F_{t,6}(k, l, f, u) &= P(U_t = u | S_t = l, F_t = f) \cdot F_{t,5}(k, l, f) \end{aligned}$$

Hence we have computed :

$$F_{t,6}(k, l, f, u) = P(Q_t = k, S_t = l, F_t = f, U_t = u | y_{1:t-1})$$

We can finally update the forward variable :

$$\alpha_t(k, l, f, u) = c_t \cdot P(y_t | Q_t = k, S_t = l) \cdot F_{t,6}(k, l, f, u)$$

with c_t a scaling coefficient defined by :

$$c_t = \sum_{k,l,f,u} \alpha_t(k, l, f, u)$$

The complete recurrence relation can be deduced :

$$\begin{aligned} \alpha_t &= c_t \cdot b_k^{(l)}(y_t) \cdot \{uf \cdot g_{l \text{ end}} + (1-u)(1-f \cdot g_{l \text{ end}})\} \\ &\quad \cdot \{f \cdot a_k^{(l)} \text{ end} + (1-f)(1-a_k^{(l)} \text{ end})\} \\ &\quad \cdot \sum_j \sum_f \{ \\ &\quad \quad [(1-f) \cdot \tilde{a}_{jk}^{(l)} + f \cdot \pi_k^{(l)}] \\ &\quad \cdot \left[\sum_i \sum_u [f[u \cdot h_l + (1-u)\tilde{g}_{il}] + (1-f)\delta(i, l)] \alpha_{t-1}(j, i, f, u) \right] \} \end{aligned}$$

A.2.2 Reduction

Considering that two nodes of the model are binary, simplifications can be achieved. In particular, the value of U_t is conditioned on F_t , because the *symbolic* level can only finish – enter its *exit* state – if and only if the *production* level has already finished. As a consequence, three values are acceptable for the couple $\{F_t, U_t\}$.

Defining $E_t = F_t + U_t \in \{0, 1, 2\}$, we can propose a new definition of the forward variable:

$$\alpha_t^e(j, i) = P(Q_t = j, S_t = i, E_t = e | y_{1:t})$$

with

$$E_t = \begin{cases} 0 & \text{if } F_t = 0 \\ 1 & \text{if } F_t = 1 \text{ and } U_t = 0 \\ 2 & \text{if } F_t = 1 \text{ and } U_t = 1 \end{cases}$$

Separating the three possible cases, important simplifications can be done, included in the algorithm of the following section.

A.2.3 Algorithm

Initialization: $t = 1$

$$\begin{aligned}\alpha_1^2(k, l) &= 0 \\ \alpha_1^1(k, l) &= 0 \\ \alpha_1^0(k, l) &= h_l \cdot \pi_k^{(l)} \cdot b_k^{(l)}(y_1)\end{aligned}$$

Propagation: $t = 2 \dots T-1$

Compute an intermediate quantity:

$$V_f(k, l) = \pi_k^{(l)} \sum_i \left[\tilde{g}_{il} \sum_j \alpha_{t-1}^1(j, i) + h_l \sum_j \alpha_{t-1}^2(j, i) \right] + \sum_j \left[\tilde{a}_{jk}^{(l)} \cdot \alpha_{t-1}^0(j, l) \right]$$

Then Update forward variable:

$$\begin{aligned}\alpha_t^2(k, l) &= b_k^{(l)}(y_t) \cdot g_{l \text{ end}} \cdot a_{k \text{ end}}^{(l)} \cdot V_f(k, l) \\ \alpha_t^1(k, l) &= b_k^{(l)}(y_t) \cdot (1 - g_{l \text{ end}}) \cdot a_{k \text{ end}}^{(l)} \cdot V_f(k, l) \\ \alpha_t^0(k, l) &= b_k^{(l)}(y_t) \cdot (1 - a_{k \text{ end}}^{(l)}) \cdot V_f(k, l)\end{aligned}$$

Scale the forward Variable:

$$\begin{aligned}c_t &= \frac{1}{\sum_{k,l,e} \alpha_t(k, l, e)} \\ \alpha_t(k, l, e) &= c_t \cdot \alpha_t(k, l, e)\end{aligned}$$

Termination: $t = T$

$$\begin{aligned}\alpha_T^2(k, l) &= c_T \cdot b_j^{(l)}(y_T) \cdot g_{l \text{ end}} \cdot a_{j \text{ end}}^{(i)} \cdot V_f(k, l) \\ \alpha_T^1(k, l) &= 0 \\ \alpha_T^0(k, l) &= 0\end{aligned}$$

Segmentation Technique:

The most probable symbolic state at each time step can be computed by:

$$\{Q_t^*, S_t^*, E_t^{(*)}\} = \underset{k,l,e}{\operatorname{argmax}} \alpha_t(k, l, e)$$

A.3 Fixed-Lag Smoothing

Fixed-Lag Smoothing is the process of estimating a state of the past given the evidence up to the current time. Defining $X_t = \{Q_t, S_t, F_t, U_t\}$ the set of nodes at time t , the smoothing operation amounts to estimate $P(X_{t-L}|y_{1:t})$ where L is a constant called lag.

The basic idea is to add a backward pass to the simple forward update of the previous section. At each time step, the forward variable is updated, and a backward operation is repeated from time t to $t - L$. We must then introduced a backward variable defined by:

$$\beta_{t-\tau}^e(j, i) = P(Q_{t-\tau} = j, S_{t-\tau} = i, E_{t-\tau} = e | y_{t-\tau+1:t})$$

The update of the backward variable derives from the backward pass of the frontier algorithm. As the process is analogous to the forward update of the previous section, we do not give details and introduce the simplified backward pass:

Initialization: $\tau = -1$

$$\beta_{t+1}^e(j, i) = 1$$

Propagation: $\tau = 0 \cdots L$

Define intermediate variable:

$$F_B(k, l) = b_k(y_t) \cdot \left\{ (1 - a_{k \text{ end}}^{(l)}) \cdot \beta_{t-\tau+1}^0(k, l) + a_{k \text{ end}}^{(l)} \cdot \left[(1 - g_{l \text{ end}}) \cdot \beta_{t-\tau+1}^1(k, l) + g_{l \text{ end}} \cdot \beta_{t-\tau+1}^2(k, l) \right] \right\}$$

Update backward variable:

$$\begin{aligned} \beta_{t-\tau}^0(j, i) &= \sum_l \delta(i, l) \sum_k \tilde{a}_{jk}^{(l)} \cdot F_B(k, l) \\ &= \sum_k \tilde{a}_{jk}^{(i)} \cdot F_B(k, i) \\ \beta_{t-\tau}^1(j, i) &= \sum_l \tilde{g}_{il} \cdot \sum_k \pi_k^{(l)} \cdot F_B(k, l) \\ \beta_{t-\tau}^2(j, i) &= \sum_l \pi_l \cdot \sum_k \pi_k^{(l)} \cdot F_B(k, l) \end{aligned}$$

Segmentation technique

At each time step, a forward update is computed, followed by a backward pass. This allow for deriving:

$$\begin{aligned} \gamma_{t-L}^e(j, i) &= \alpha_{t-L}^e(j, i) \cdot \beta_{t-L}^e(j, i) \\ &= P(Q_{t-L} = j, S_{t-L} = i, E_{t-L} = e | y_{1:t}) \end{aligned}$$

Finally, the Fixed-lag smoothing algorithm can be summarized in pseudo-code by:

```

t = 1:
  alpha[1] = init_forw(y[1])

FOR t = 2:∞
  alpha[t] = forward(y[t] , alpha[t-1])
  IF t >= L

    beta = 1
    FOR k = 0:L DO
      beta = backward(y[t-k] , beta)
    END FOR
    gamma[t-k] = alpha[t-k].*beta
    state[t-k] = argmax(gamma[t-k])

  END IF
END FOR

```

A.4 Viterbi Algorithm

The algorithm for the forward Viterbi procedure is similar to the forward algorithm, replacing sums by maximizations.

Let $X_{1:t} = \{Q_{1:t}, S_{1:t}, F_{1:t}, U_{1:t}\}$, define the Dynamic programming variable :

$$\delta_t(j, i, f, u) = \max_{X_{1:t-1}} P(Q_t = j, Q_{1:t-1}, S_t = i, S_{1:t-1}, F_t = f, F_{1:t-1}, U_t = u, U_{1:t-1}, y_{1:t})$$

The variable is updated at each time step by :

$$\begin{aligned} \delta_t(k, l, f, u) = & b_k^{(l)}(y_t) \cdot \{uf \cdot g_{l \text{ end}} + (1 - u)(1 - f \cdot g_{l \text{ end}})\} \\ & \cdot \{f \cdot a_{k \text{ end}}^{(l)} + (1 - f)(1 - a_{k \text{ end}}^{(l)})\} \\ & \cdot \max_{j, f_0} \left\{ (1 - f_0) \cdot \tilde{a}_{jk}^{(l)} + f_0 \cdot \pi_k^{(l)} \right\} \\ & \cdot \max_{i, u_0} \{ [f_0 [u_0 \cdot h_l + (1 - u_0) \tilde{g}_{il}] + (1 - f_0) \delta(i, l)] \delta_{t-1}(j, i, f_0, u_0) \} \end{aligned}$$

A.4.1 Algorithm : Viterbi Decoding

Initialization: $t = 1$

$$\begin{aligned} \delta_1^2(k, l) &= 0 \\ \delta_1^1(k, l) &= 0 \\ \delta_1^0(k, l) &= h_l \cdot \pi_k^{(l)} \cdot b_k^{(l)}(y_1) \end{aligned}$$

$$\Psi_1(k, l) = \{0, 0, 0\}$$

Propagation: $t = 2 \dots T-1$

$\forall k, l, j, i, e$, Compute V_0^e :

$$\begin{aligned} V_0^2(k, l, j, i) &= \pi_k^{(l)} \cdot h_l \cdot \delta_{t-1}^2(j, i) \\ V_0^1(k, l, j, i) &= \pi_k^{(l)} \cdot \tilde{g}_{il} \cdot \delta_{t-1}^1(j, i) \\ V_0^0(k, l, j, i) &= \tilde{a}_{jk}^{(l)} \cdot \delta(i, l) \cdot \delta_{t-1}^0(j, i) \end{aligned}$$

Maximize over variables at previous time step:

$$\begin{aligned} V_1(k, l) &= \max_{j, i, e} V_0^e(k, l, j, i) \\ \Psi_t(k, l) &= \operatorname{argmax}_{j, i, e} V_0^e(k, l, j, i) \end{aligned}$$

Then Update forward variable: $\forall k, l$,

$$\begin{aligned} \delta_t^2(k, l) &= b_k^{(l)}(y_t) \cdot g_{l \text{ end}} \cdot a_{k \text{ end}}^{(l)} \cdot V_1(k, l) \\ \delta_t^1(k, l) &= b_k^{(l)}(y_t) \cdot (1 - g_{l \text{ end}}) \cdot a_{k \text{ end}}^{(l)} \cdot V_1(k, l) \\ \delta_t^0(k, l) &= b_k^{(l)}(y_t) \cdot (1 - a_{k \text{ end}}^{(l)}) \cdot V_1(k, l) \end{aligned}$$

Termination: $t = T$

$$\begin{aligned}\delta_T^2(k, l) &= b_k^{(l)}(y_T) \cdot g_{l_end} \cdot a_k^{(l)} \cdot V_1(k, l) \\ \delta_T^1(k, l) &= 0 \\ \delta_T^0(k, l) &= 0\end{aligned}$$

Backtracking

Define the state sequence $X_t^* = \{Q_t^*, S_t^*, F_t^*, U_t^*\}$

$$\begin{aligned}P^* &= \max_{j,i,f,u} \delta_T^2(k, l) \\ X_T^* &= \operatorname{argmax}_{j,i,f,u} \delta_T^2(k, l)\end{aligned}$$

Trace back to find the optimal state sequence:

for $t = T-1 \dots 1$:

$$X_t^* = \Psi_{t+1}(X_{t+1}^*)$$

Finally, the optimal state sequence is obtained, giving at each time step the *symbolic* state, i.e. the most probable primitive gesture and the *production* state which permits an analysis of the timing execution of the gesture in comparison with the original template.