# Saxophone improvisé: modélisation temporelle et segmentation en temps réel

Rapport de stage M2 ATIAM
**Ircam — Université Paris 6**
*Equipe Applications Temps Réel*
*Sous la direction de Norbert Schnell*

Pierre DUQUESNE

Juin 2006

# Table des matières

# Table des figures

Un immense merci à Julien Bloit pour m’avoir guidé tout au long du stage.

Un grand merci à Norbert Schnell pour m’avoir accepté dans son équipe.

Un grand merci à Arshia Cont pour ses conseils précieux.

# Introduction

Ce travail a été effectué dans le cadre de l'utilisation de l'ordinateur dans l'improvisation. Après avoir présenté les techniques de modélisation probabilistes du signal utilisant les modèles de Markov cachés, nous nous intéressons successivement à trois problèmes. Premièrement, la modélisation des événements musicaux du saxophone, en utilisant seulement les descripteurs de Yin. Ensuite, l'identification de ces événements dans l'improvisation. Enfin, en vue d'une application en temps-réel, la réalisation de cette tâche grâce à des algorithmes qui utilisent les données de manière causale, au fur et à mesure qu'elles sont reçues.

La motivation de cette recherche est de comparer les résultats d'une modélisation probabiliste avec un algorithme existant de segmentation qui utilise uniquement Yin, et construit à partir d'heuristiques.

# Chapitre 1

# Travaux précédents

Les phénomènes non-déterministes, comme le lancer de dés, suivent des lois probabilistes. On distingue la probabilité à priori $P(x)$ — qui est la probabilité d'obtenir par exemple la valeur $x = 10$ en lançant deux dés — de la probabilité à posteriori $P(x|y)$ — probabilité d'obtenir $x = 10$ sachant que le premier dé a donné $y = 4$. Il est possible d'estimer la loi de probabilité régissant un système en observant un grand nombre de réalisations. La loi de probabilité d'un dé normal n'est pas la même que celle d'un dé pipé. Cette propriété rend possible de deviner à quelle catégorie (pipé/non-pipé) appartient un dé en observant uniquement les valeurs qu'il donne. Ranger en catégorie est ce que l'on appelle la classification.

Le théorème de Bayes $P(k|d) = \frac{P(d|k)\,P(k)}{P(k|d)}$ est à l'origine de l'inférence bayesienne, une méthode probabiliste développée en intelligence artificielle permettant d'augmenter les certitudes que l'on a sur un système grâce à ce que l'on observe. Un modèle de Markov caché est un cas particulier de réseau bayésien, le support théorique de l'inférence bayésienne.

Dans ce chapitre, nous présenterons la théorie des modèles de Markov, et leur utilisation pour l'analyse des signaux sonores en reconnaissance de la parole et pour le suivi de partition est décrite. Puis les descripteurs calculés par l'algorithme Yin sont présentés.

## 1.1 Modèles de Markov cachés

## 1.2 Présentation

Les modèles graphiques allient la théorie des probabilités et la théorie des graphes. Ce sont des graphes dont les nœuds représentent des variables aléatoires. Les nœuds sont reliés par des arcs. Lorsque les arcs sont dirigés, on parle de réseau bayésien ; un réseau bayésien dynamique modélise un processus stochastique. Les modèles de Markov cachés (MMC), en anglais *Hidden Markov Models* (HMM) sont des cas particuliers de réseaux bayésiens. Ils ont été introduits par Baum *et al* à la fin des années 60 afin de modéliser des séquences d'observations. Les MMC ont été adoptés pour la modélisation du signal en reconnaissance de la parole.

## 1.3 Définition

Un modèle de Markov caché est un automate à états discrets non-déterministe génératif. Il est défini par :

- Un ensemble de $N$ états : $S = \{s_1, s_2, ..., s_N\}$
- La distribution de probabilité de transitions entre états : $A = \{a_{ij}\}$
- La distribution conditionnelle d'observation : $B = \{b_j(O_t) = P(O_t|s_j)\}$

– La probabilité initiale de chaque état : $\pi = \{\pi_i\}$

Certaines probabilités de transition peuvent être nulles. L'ensemble des transitions non-nulles définit la topologie d'un modèle (figure 1.1) . On parle de modèle caché quand on n'observe pas directement les états, mais seulement les valeurs générées. On fait alors l'hypothèse que le signal est un processus markovien et on cherche à estimer les paramètres du modèle caché $\lambda = (A, B, \pi)$.

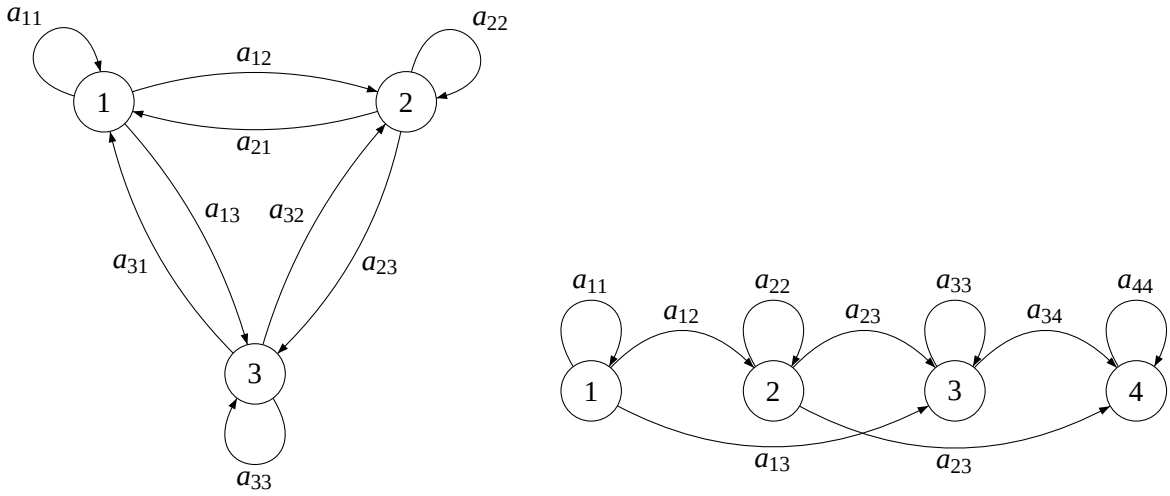


FIG. 1.1: Deux topologies différentes. (a) un modèle ergodique à 3 états, (b) un modèle gauche-droite à 4 états.

Un MMC peut-être utilisé comme un générateur d'observations $O = O_1, O_2, \ldots, O_T$ de la façon suivante :

– Choisir un état initial $q_i = s_i$ en respectant la distribution $\pi$.

– Pour chaque $t$ :

Emettre $O_t$ en respectant la distribution de probabilité de l'état $b_i$ de $s_i$.

Sauter vers un autre état $q_{t+1} = s_j$ en respectant la distribution de probabilité $a_{ij}$

La séquence d'état associée à $O$ est $Q = q_1, q_2, \ldots, q_T$.

## 1.4 Les trois problèmes des MMC et leur solutions.

Les trois problèmes classiques qu'on cherche à résoudre, étant donnée une séquence d'observation $O = O_1, O_2, \ldots, O_T$ sont les suivants :

### 1.4.1 Problème 1 : Calcul de la vraisemblance

Comment calculer efficacement la probabilité (ou vraisemblance) $P(O|\lambda)$ de la séquence d'observation étant donné le modèle ? Une solution efficace est d'utiliser la procédure forward-backward :

1. Initialisation

$$\alpha_1(i) = \pi_i b_i(O_1) \qquad 1 \leq i \leq N \tag{1.1}$$

2. Induction

$$\alpha_{t+1}(j) = [\sum_{j=1}^{N} \alpha_t(i) a_{ij}] b_j(O_{t+1}) \quad 1 \leq t \leq T-1; \;\; 1 \leq j \leq N \tag{1.2}$$

3. Terminaison

$$P(O|\lambda) = \sum_{i=1}^{N} \alpha_T(i) \tag{1.3}$$

La variable forward $\alpha$ est suffisante pour calculer la vraisemblance d'un séquence d'observation. La variable backward ($\beta$) est utilisée pour résoudre le problème 3 par l'algorithme de Baum-Welch. Une interprétation plus intuitive de la variable $\alpha$ est présentée en 2.3.2.

### 1.4.2 Problème 2 : Séquence d'état optimale

Comment trouver la séquence d'état optimale $Q = q_1, \ldots, q_T$ ? Ce problème est résolu par l'algorithme de Viterbi :

1. Initialisation

$$\delta_1(i) = \pi_i b_i(O_1), \quad 1 \leq i \leq N \tag{1.4}$$
$$\psi_t(j) = 0 \tag{1.5}$$

2. Recursion

$$\delta_t(j) = \max_{1 \leq i \leq N} [\delta_{t-1}(i) a_{ij}] b_j(O_t), \quad 2 \leq t \leq T, \ 1 \leq j \leq N \tag{1.6}$$
$$\psi_t(j) = \operatorname*{argmax}_{1 \leq i \leq N} [\delta_{t-1}(i) a_{ij}], \quad 2 \leq t \leq T, \ 1 \leq j \leq N \tag{1.7}$$

3. Terminaison

$$q_T^* = \operatorname*{argmax}_{1 \leq i \leq N} [\delta_T(i)] \tag{1.8}$$

4. Trouver la séquence d'état (backtracking)

$$q_t^* = \psi_{t+1}(q_{t+1}^*), \quad t = T-1, T-2, \ldots, 1 \tag{1.9}$$

Une interprétation plus intuitive des variables $\delta$ et $\psi$ est présentée en 2.3.2.

### 1.4.3 Problème 3 : Estimer les paramètres du modèle

Comment estimer les paramètres $\lambda = (A, B, \pi)$ pour maximiser $P(O|\lambda)$ ? Ce problème peut être résolu par l'algorithme de Baum-Welch ou bien par l'algorithme segmental k-means. Ce dernier est décrit en ref page ref.

## 1.5 Estimation des densités de probabilité des variables aléatoires associées aux états

Pour estimer la densité de probabilité d'une variable aléatoire, on peut faire l'hypothèse qu'elle suit une loi exprimée sous la forme d'une fonction paramétrique. Une variable aléatoire continue $X$ obéit à une loi normale $\mathcal{N}$ de paramètres $\mu$ et $\sigma^2$, si sa distribution de probabilité est donnée par la courbe délimitée par la fonction de densité :

$$f(x) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{1}{2}(\frac{x-\mu}{\sigma})^2}$$

L'estimation consiste alors à mesurer la moyenne $\mu$ et la variance $\sigma^2$ des tirages de la variable aléatoire. On estime souvent les distributions $b_j$ des modèles de Markov cachés par des mélanges de $M$ gaussiennes (somme pondérée de lois normales) multidimensionnelles :

$$f(x) = \sum_{i=1}^{M} w_i f_i(x)$$

où $w_i$ sont les poids tels que $\sum_{i=1}^{M} w_i = 1$. L'algorithme EM (Expectation Maximization) permet d'estimer les paramètres d'un mélange de gaussiennes.

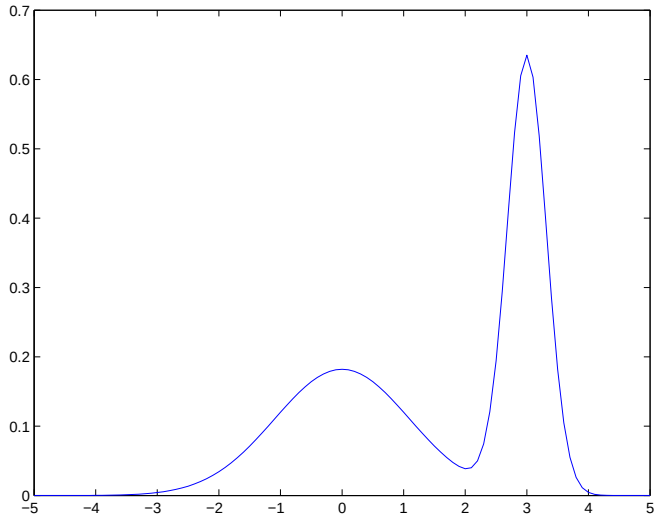


FIG. 1.2: Exemple de mélange de gaussiennes à une dimension ($\mu_1 = 0$; $\sigma_1^2 = 1,2$; $w_1 = 0.5$; $\mu_2 = 3$; $\sigma_2^2 = 0,1$; $w_2 = 0.5$ )

## 1.6 Utilisation des MMC pour l'analyse du signal sonore

Ils ont été utilisés très tôt pour la modélisation du signal en reconnaissance de la parole, car ils sont un modèle de signal aléatoire non-stationnaire, ie. dont les propriétés varient dans le temps. L'apprentissage consiste à déterminer les paramètres $\lambda$ du modèle à partir de plusieurs exemples.

### 1.6.1 Reconnaissance de la parole

Publié en 1989, le tutoriel de L. Rabiner [Rab90] fait référence en traitement du signal. Il y décrit la théorie des MMC puis ses applications pour la reconnaissance de la parole. Il faut retrouver dans le signal les unités de parole, qui peuvent être (au choix) des phonèmes, diphones, syllabes ou autres. En reconnaissant une séquence de ces unités, on peut reconstituer des mots puis des phrases. Chaque unité est représentée par un modèle de Markov caché et les mots sont construits en concaténant ces derniers. La classification de mots isolés consiste à trouver le modèle qui maximise la vraisemblance des observations. La taille du vocabulaire peut atteindre plusieurs milliers de mots.

La reconnaissance des mots dans une phrase complète fait appel à des techniques différentes car il n'est pas envisageable d'avoir un modèle par phrase étant donné le nombre de combinaisons

possibles. La lourdeur combinatoire de la reconnaissance a fait émerger des algorithmes réduisant la complexité des calculs, tout en éliminant le maximum d'hypothèses fausses. Par exemple, les règles de grammaire sont modélisées et imposent une structure à l'ordre des mots. Les réseaux de reconnaissance cherchent l'enchainement des mots de façon à ce qu'il respecte un modèle de langage. Voir [Pau92], [OVWY94].

Des modèles gauche-droite (voir figure 1.1b) sont classiquement utilisés pour les phénomènes temporels dont les propriétés changent au cours du temps. Le modèle est visité de gauche à droite et aucun retour vers un état précédent n'est possible. Il est possible de rester dans l'état courant ou de passer à un état suivant. La probabilité d'auto-transition $a_{ii}$ exprime la durée $D$ d'un état. $D$ est une variable aléatoire dont la densité de probabilité est (figure 1.3) :

$$\mathrm{P}(D = d) = a_{ii}^{d-1} * (1 - a_{ii}) \quad \text{avec} \quad d \geq 1$$

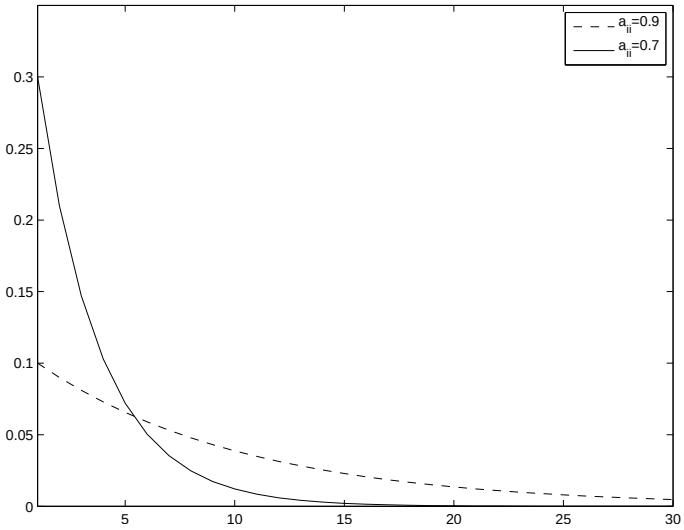


FIG. 1.3: Densité de probabilité de la durée d'un état pour $a_{ii} = 0.9$ et $a_{ii} = 0.7$

### 1.6.2 Les modèles de Markov et le signal musical

Pour réaliser un système d'accompagnement automatique, C. Raphaël [Rap98] s'intéresse à la modélisation probabiliste du signal musical. Il introduit un modèle de note, gauche-droite, à 3 états : attaque, tenue et silence. Le nombre d'état tenue peut être augmenté pour représenter une note plus longue. Pour chaque note de la partition, la distribution de probabilité associée aux états est fixée en fonction de ce qui est attendu, en particulier la hauteur. Les modèles sont ensuite reliés pour former le modèle de partition qui est entrainé par l'algorithme de Baum-Welch. La technique de segmentation en temps-réel qu'il développe — et qui consiste à maximiser la probabilité à postériori de la segmentation plutôt que celle de la séquence d'état — est fondée sur la connaissance préalable de la partition.

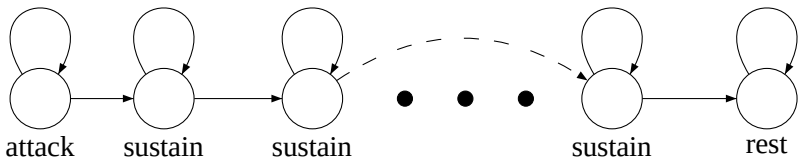


FIG. 1.4: Modèle de note

Pour le suivi de l'Ircam, le modèle de note a été étendu [Sch04] pour représenter la diversité regroupée sous le terme de note. Il possède deux états attaque, le second étant emprunté lors d'un legato. L'état silence peut être sauté quand une note précède un legato (voir figure 1.5). L'entrainement est fait grâce à une supervision automatique. L'enregistrement audio de la pièce est segmenté grâce à un algorithme basé sur des heuristiques [Con04], sans modélisation statistique ni apprentissage. Puis chaque note est isolée et entrainée séparément. Enfin, le modèle de partition est entrainé dans son ensemble.

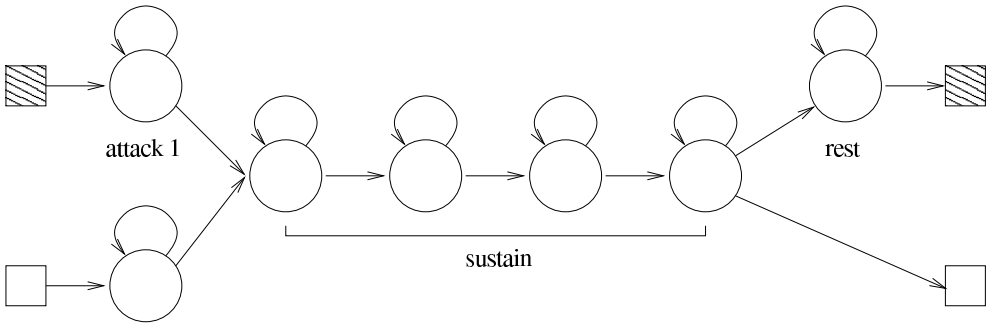


FIG. 1.5: Modèle de note pour le suivi [Sch04]

## 1.7 Descripteurs : Yin

Les observations ne sont pas les échantillons du signal, mais des descripteurs calculés sur une fenêtre de temps. Il existe beaucoup de descripteurs, comme LPC (Linear Prediction Coefficients) ou MFCC (Mel Frequency Cepstral Coefficients). Dans le cadre de ce stage, on se propose d'utiliser uniquement un ensemble réduit de descripteurs, ceux calculés par l'algorithme Yin. Ce choix est motivé par les bons résultats obtenus par l'algorithme décrit dans (cite AssayagChemillier), basé sur des heuristiques, et l'envie de comparer ces résultats avec une modélisation statistique.

Yin est un algorithme d'estimation de la fréquence fondamentale proposé par de Cheveigné et Kawahara [dCK02]. Il extrait d'un son, à des intervalles de temps réguliers, 3 descripteurs :

- la fréquence fondamentale
- le degré de périodicité
- la puissance

Yin se base sur la méthode de l'autocorrélation et permet de gérer diverses formes d'apériodicité. Sans décrire l'algorithme, j'expose les paramètres qui doivent être réglés pour obtenir de bons résultats :

- Fréquences limites de la hauteur : un choix judicieux pour les fréquences minimale et maximale qu'on souhaite détecter améliorent les résultats.

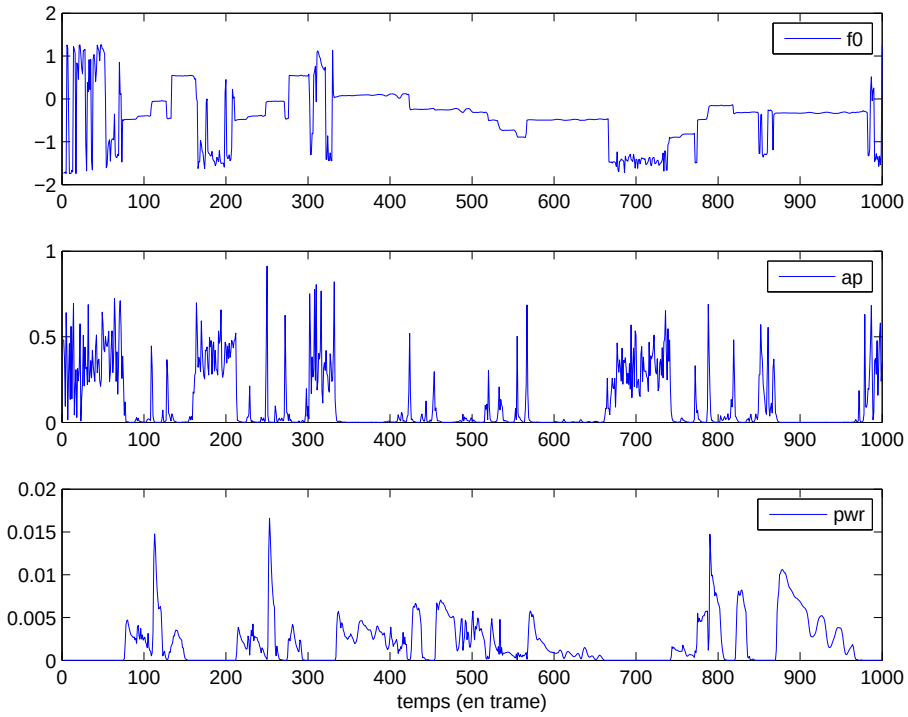


FIG. 1.6: Fréquence fondamentale (en octave relativement au La 440), apériodicité et puissance calculées par Yin, sur 13 secondes d'une improvisation de saxophone

- Taille de la fenêtre : L'intervalle de temps sur lequel se fait l'analyse (trame).
- Hopsize : Les trames se chevauchent dans le temps, le hopsize est l'intervalle entre deux débuts de fenêtre (inférieur à la taille de la fenêtre)

Contrairement à d'autres méthodes, la taille de la fenêtre et la fréquence minimale sont découplées. Mais pour avoir une estimation stable, la longueur de la fenêtre doit être au moins longue comme deux fois la plus grande période qu'on cherche à détecter. La fenêtre peut être plus grande, et on observe en pratique une meilleure estimation.

# Chapitre 2

# Travail réalisé

Comme point de départ, nous disposions d'un enregistrement d'une improvisation de saxophone de cinq minutes environ. Le premier travail a été d'identifier les types d'événements musicaux présents dans cet enregistrement. Ce qui est entendu par événement est une unité du discours musical. Il s'agit de notes ordinaires et de silence, bien sûr, mais aussi de glissandi, vibratos, sons décoloré ou éoliens, multiphoniques, trilles. Ces modes de jeu (terme pris dans un sens assez large) font partie de la palette expressive de l'improvisateur. Dans le cadre d'une recherche sur l'interaction musicien/ordinateur pour la musique improvisée, il est souhaitable de pouvoir construire une description symbolique de haut niveau du signal sonore qui intègre la richesse du langage improvisé. Cette description permettra de découper et d'isoler les unités du discours, et éventuellement les recombiner. De plus, pour autoriser l'ordinateur à prendre part à l'improvisation on veut que cette description soit construite en temps-réel. Les unités doivent être identifiées au fur et à mesure qu'elles sont jouées.

La deuxième étape a été de repérer manuellement dans l'enregistrement les éléments et leur classe associée. Les segments isolés seront utilisés comme exemples pour l'apprentissage statistique, et les données temporelles comme référence pour l'évaluation de la segmentation/classification. La segmentation manuelle n'est en aucun cas parfaite. Il est difficile – en particulier dans le cas du saxophone qui permet des attaques douces et des notes liées – de déterminer avec une certitude absolue à quel instant commence un événement et quand il finit. Il arrive qu'à l'écoute d'une phrase musicale, on discerne clairement 8 notes. Or au moment de les délimiter, on ne parvient à en retrouver que sept. La huitième note fantôme, fait bien partie du discours musical, mais pas du signal sonore. Son existence est dûe à des processus cognitifs de haut niveau (top-down), et de notre entrainement à l'écoute musicale. De même, attribuer une classe est diffile car il n'y a pas toujours de limites nettes entre les catégories. Il existe par exemple un univers continu de variations entre un son décoloré – dans lequel le souffle est très présent et le son détimbré – et un son ordinaire. Un son peut-être plus ou moins décoloré, et il est impossible de décider sur des critères objectifs dans quelle catégorie le ranger. La segmentation/classification manuelle est forcément subjective.

La suite du travail se divise en trois parties successives. Tout d'abord, le choix des modèles et leur entrainement. Puis un modèle pour la segmentation/classification est progressivement construit. Enfin, l'application en temps-réel est envisagée. Tous les résultats présentés ont été obtenus sur un ensemble n'ayant pas servi à l'apprentissage.

## 2.1 Modèles isolés

### 2.1.1 Entrainement des modèles

Une fois les différentes classes déterminées, le fichier son segmenté et annoté, il faut créer et entrainer les modèles des éléments. Nous nous restreignons dans un premier temps à deux classes basiques : note et silence. L'objectif est de pouvoir étendre les classes par la suite, en introduisant de nouveaux modèles. Les modèles utilisés sont des modèles de Markov dont les densités de probabilité associées aux états sont de simples gaussiennes. Ce choix est déterminé par une simplicité d'implémentation mais aussi par une démarche consistant à partir de choses simples et à les étendre progressivement, afin de bien comprendre les effets des modifications. Pour l'entrainement des modèles, nous avons implémenté l'algorithme segmental k-means décrit dans [Rab90], et dont la convergence est démontrée en 1990 [JR90]. Etant donnée une topologie de modèle, il consiste à :

1. Initialiser les états de manière aléatoire. En pratique nous avons initialisé les états par clustering [1] sur les descripteurs.
2. Pour chaque exemple, décoder la séquence d'état optimale par l'algorithme de Viterbi
3. Réestimer les lois des états en fonction du décodage.
4. Réestimer les probabilités de transition suivant la formule :

$$\bar{a_{ij}} = \frac{||T_{ij}||}{\sum_{j=i}^{N} ||T_{ij}||}, \quad i,j = 1,2,3...,N \tag{2.1}$$

   où $N$ est le nombre d'états et $T_{ij}$ est l'ensemble des passages de l'état $i$ à l'état $j$ dans les décodages.
5. Réestimer les probabilités initiales : $\bar{pi_i}$ est la proportion de $s_i$ décodés comme premier état.
6. Recommencer à l'étape 2.

A chaque itération, la vraisemblance du modèle augmente. La décision de stopper l'entrainement est prise lorsque l'augmentation de la vraisemblance entre deux itérations est inférieure à un seuil.

### 2.1.2 Choix des modèles

Nous souhaitons construire de bons modèles de nos évènements musicaux. Qu'est-ce qu'un bon modèle ? Cela dépend de la tâche pour laquelle ils sont utilisés. Nous souhaitons que nos modèles permettent :

- d'identifier la classe
- de connaître les états de début et de fin, pour pouvoir segmenter.

Dans son tutoriel de 1989, Rabiner explique qu'il n'existe pas de solution théorique pour choisir un modèle. Ce sujet de recherche est toujours très actif. Il y a deux niveaux sur lesquels nous pouvons agir, ce sont la topologie du modèle et les descripteurs utilisés pour décrire le signal sonore.

#### Topologie

Dans un premier temps, nous avons entrainé sur des exemples isolés des modèles ergodiques (voir figure 1.1 p. 7) à trois états, avec distribution de probabilité initiale homogène. Nous avons entrainé en un pour la classe silence, et un pour la classe ordinario (note ordinaire), avec une combinaison assez quelconque de descripteurs Yin. Les performances de classification (pourcentage

[1] Technique pour identifier des régions dans un espace vectoriel

d'éléments isolés bien classés) se sont révélées bonnes, cependant les séquences d'états décodées n'ont pas permis de discerner des états initiaux ou finaux.

C'est pourquoi nous avons ensuite testé des modèles gauche-droite à 3 états, avec le premier état imposé par la distribution de probabilité initale. C'est le modèle classique de note, avec un seul état pour le sustain. Les performances de classification ont été un peu diminué, mais sont restées acceptables (voir 2.1.2 pour les chiffres).

Bien que l'état initial soit maintenant imposé, rien n'impose que l'état final soit le troisième. Le décodage du silence, par exemple, a montré que seul les deux premiers états du modèle étaient utilisés. C'est pourquoi le modèle silence a été réduit à deux états. Une interprétation de ce constat est donnée en 2.1.3.

Pour le modèle de note, le constat après entrainement est que le deuxième et le troisième état ne sont pas toujours atteints. Certains décodages n'utilisent qu'un ou deux états. Pour remédier à cela, le choix des descripteurs a été effectué de manière à maximiser la réussite de la classification et la proportion des décodages atteignant le troisième état.

**Descripteurs**

Les trois descripteurs utilisés sont calculés par l'algorithme Yin. Ce sont la fréquence fondamentale, l'apériodicité et l'énergie. En statistique il faut souvent faire des prétraitements sur les données. Plutôt que prendre la fréquence en Hertz, nous utilisons son logarithme afin d'unifier la valeur de l'octave, et coller à la perception. Le descripteur de la hauteur n'est pas utilisé directement car il n'est pas immédiatement pertinent pour l'identification des modes de jeu. En effet, on souhaite reconnaitre une classe quelque soit sa hauteur. Par contre, on s'intéresse à la variation de la hauteur dans le temps. Elle peut par exemple permettre l'identification du vibrato et du glissando. Nous choisissons donc d'utiliser la dérivée discrète de la hauteur dans nos modèles.

Pendant les phases de silence, les valeurs de hauteur et d'apériodicité prennent des valeurs quasi-aléatoires (voir figure 1.7 page 12). C'est pourquoi nous avons fait l'essai de forcer leur valeur quand la puissance est inférieure à un certain seuil. Cependant, ce traitement n'a pas été concluant, dégradant considérablement la classification. Il se trouve que le caractère bruité de ces descripteurs, avec une large répartition des valeurs, pendant les phases de silence est – assez paradoxalement – informatif.

Enfin, il est aussi possible de dériver les descripteurs apériodicité et puissance. La sélection des descripteurs s'est faite par sélection progressive des descripteurs pertinents.

Deux critères ont été utilisés pour choisir une bonne combinaison de descripteurs :

1. Le pourcentage de réussite pour la reconnaissance : pour attribuer une classe à une séquence d'observation, la séquence d'état optimale est calculée pour chaque modèle. Leur vraisemblance sont comparées et la classe choisie correspond au modèle pour lequel la vraisemblance est maximale.

2. Le pourcentage d'état final atteint : Chaque séquence de test est décodée avec le modèle de sa classe, et la proportion de décodages atteignant le dernier état est calculé.

Pour éviter de tester toutes les combinaisons, la selection s'est faite sur un, puis deux, puis trois descripteurs, en en éliminant à chaque étape. Les statistiques sont présentées dans le tableau 2.1. Remarquons qu'ajouter des descripteurs n'améliore pas systématiquement les résultats. Il est difficile de faire correspondre deux descripteurs décorrélés aux mêmes états d'un modèle. Nous avons choisi la combinaison puissance, dérivée de l'apériodicité, dérivée de la hauteur pour ses bons résultats.

Le *hopsize* a été fixé à 500 échantillons, et la fenêtre à 1024, de manière empirique. La fréquence minimale a été fixée au Do 2 (132Hz) et la fréquence maximale au Do4 (1056Hz).

| % | ord | sil | Total |
|---|---|---|---|
| Ap | 96.3 | 91.2 | 93.8 |
| Pwr | 87.3 | 97.1 | 92.2 |
| d1Ap | 96.6 | 86.9 | 91.7 |
| d1F0 | 94.1 | 88.3 | 91.2 |
| d1Pwr | 90.7 | 80.3 | 85.5 |
| F0 | 86.6 | 72.3 | 79.4 |

| % | ord | sil | Total |
|---|---|---|---|
| d1Ap | 66.6 | 99.3 | 82.9 |
| d1Pwr | 58.3 | 99.3 | 78.8 |
| d1F0 | 89.8 | 64.2 | 77.0 |
| Ap | 54.4 | 82.5 | 68.4 |
| Pwr | 38.8 | 94.2 | 66.5 |
| F0 | 30.2 | 66.4 | 48.3 |

| | | ord | sil | Total |
|---|---|---|---|---|
| Pwr | d1F0 | 98.5 | 96.4 | 97.4 |
| Pwr | d1Ap | 95.4 | 97.1 | 96.2 |
| d1Pwr | Ap | 95.9 | 96.4 | 96.1 |
| Pwr | Ap | 95.1 | 97.1 | 96.1 |
| Pwr | d1Pwr | 96.3 | 95.6 | 96.0 |
| d1Pwr | d1Ap | 95.4 | 96.4 | 95.9 |
| d1F0 | d1Pwr | 96.1 | 95.6 | 95.9 |
| d1F0 | d1Ap | 95.9 | 87.6 | 91.7 |
| d1F0 | Ap | 98.5 | 81.0 | 89.8 |

| | | ord | sil | Total |
|---|---|---|---|---|
| d1F0 | d1Pwr | 71.0 | 99.3 | 85.1 |
| Pwr | d1F0 | 68.3 | 94.2 | 81.2 |
| Pwr | d1Ap | 68.0 | 94.2 | 81.1 |
| d1Pwr | d1Ap | 56.6 | 99.3 | 77.9 |
| Pwr | d1Pwr | 58.8 | 94.9 | 76.8 |
| d1Pwr | Ap | 52.4 | 97.1 | 74.8 |
| Pwr | Ap | 52.0 | 92.7 | 72.3 |
| d1F0 | d1Ap | 46.1 | 67.2 | 56.6 |
| d1F0 | Ap | 34.1 | 56.9 | 45.5 |

| | | | ord | sil | Total |
|---|---|---|---|---|---|
| Pwr | d1F0 | Ap | 98.8 | 95.6 | 97.2 |
| Pwr | d1Ap | d1F0 | 98.8 | 95.6 | 97.2 |
| d1F0 | d1Ap | d1Pwr | 96.6 | 95.6 | 96.1 |
| Pwr | d1Pwr | Ap | 96.3 | 95.6 | 96.0 |
| d1Pwr | d1F0 | Ap | 95.6 | 95.6 | 95.6 |

(a)

| | | | ord | sil | Total |
|---|---|---|---|---|---|
| d1F0 | d1Ap | d1Pwr | 73.7 | 99.3 | 86.5 |
| Pwr | d1Ap | d1F0 | 65.9 | 94.9 | 80.4 |
| d1Pwr | d1F0 | Ap | 56.1 | 97.8 | 77.0 |
| Pwr | d1Pwr | Ap | 57.6 | 94.9 | 76.2 |
| Pwr | d1F0 | Ap | 48.8 | 92.7 | 70.7 |

(b)

FIG. 2.1: Evaluation des descripteurs : (a) Pourcentage de classification réussie et (b) Pourcentage d'état final atteint

### 2.1.3 Présentation des modèles entrainés

Les figures 2.2 et 2.3 présentent les distributions de probabilité associées à chaque descripteurs pour les états des modèles de note et de silence. On peut interpréter ces modèles et valider intuitivement leur pertinence.

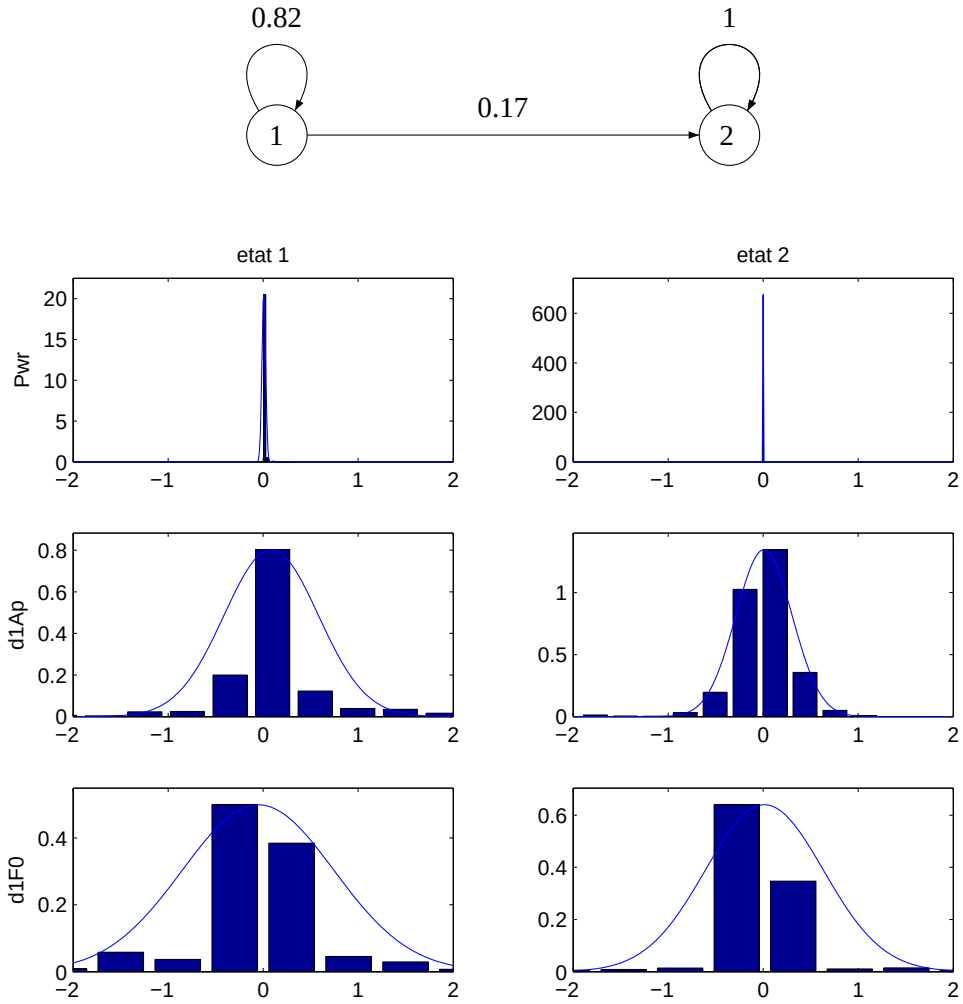


FIG. 2.2: Modèle de silence entrainé : distribution de probabilité estimée et histogramme pour les 3 descripteurs : puissance, dérivée de l'apériodicité et dérivée de la hauteur ($\Delta \log(F_O)$). La puissance est faible est les deux autres descripteurs varie aléatoirement.

Commençons par le modèle de silence. Dans les exemples donnés pour l'apprentissage, qui ont été segmentés à la main, le silence commence souvent alors que la réverbération de la note précédente se fait encore entendre. On peut observer que cela a bien été intégré dans le modèle en remarquant que dans le premier état, l'énergie est très faible et quasi-nulle dans le second. Les dispersions assez larges des dérivées de l'apériodicité et de la hauteur dans les deux états traduisent les variations quasi-aléatoires de l'apériodicité et de la hauteur pendant les silences.

Pour le modèle de note, on peut dire qu'on a retrouvé les états classiques (voir 1.6.2). Dans le premier état, la puissance est faible, signe d'une attaque progressive, typique du saxophone. Les

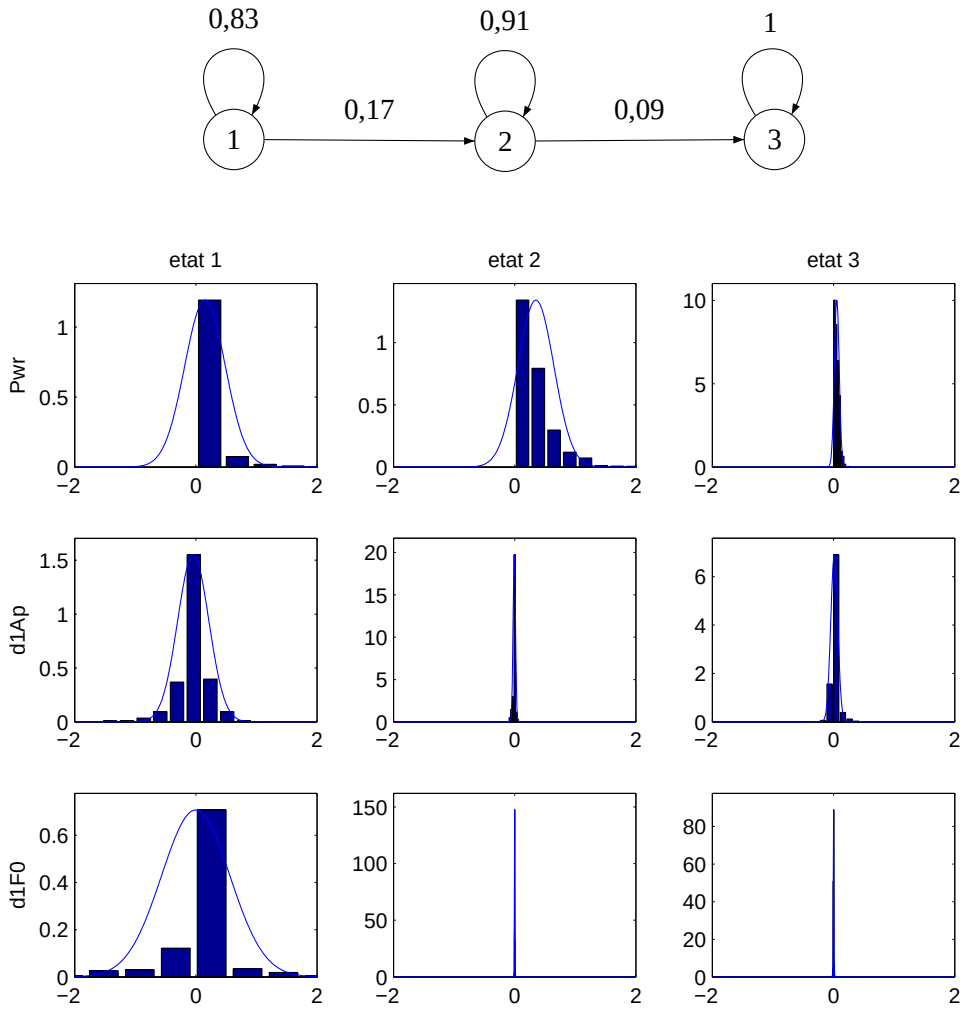


FIG. 2.3: Modèle de note entrainé : distribution de probabilité estimée et histogramme pour les 3 descripteurs : puissance, dérivée de l'apériodicité et dérivée de la hauteur ($\Delta\log(F_O)$). On peut distinguer les états attaque tenue et fin.

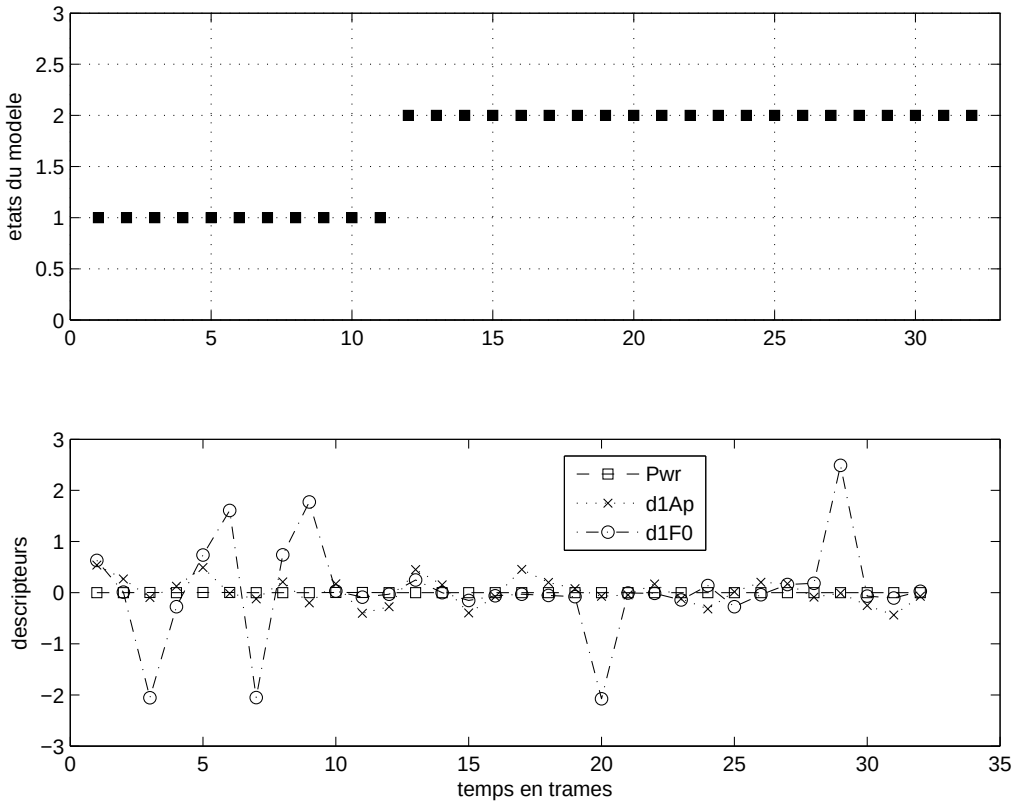


FIG. 2.4: Décodage d'un silence isolé : la puissance est quasi-nulle, la hauteur et l'apériodicité varient aléatoirement, ce qui s'observe sur leur dérivée.

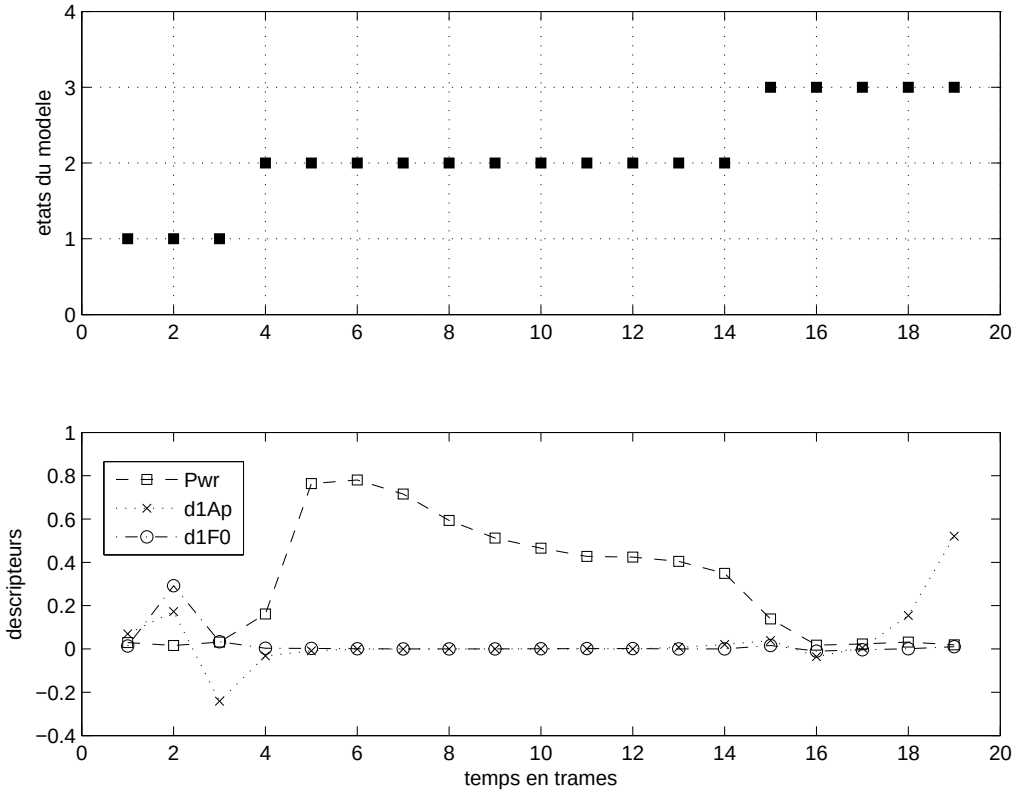


FIG. 2.5: Décodage d'une note isolée : une attaque douce avec variation de la hauteur et de l'apériodicité, puis une hauteur tenue et stable une puissance importante décroissante, et la fin de la note avec une chute de la puissance et une variation de l'apériodicité.

dérivées de la hauteur et l'apériodicité présentent une dispersion importante. En début de note, il y a un changement de hauteur brusque (après une note ou un silence). qui peut aussi expliquer la variation forte de l'apériodicité. Cela peut aussi traduire une apériodicité liée à l'attaque. Dans le deuxième état, la puissance est plus forte, la variation de hauteur quasi-nulle, l'apériodicité stable (dérivée quasi-nulle) : La note est tenue. Dans le troisième état, la fin de la note, l'apériodicité varie légèrement, la hauteur reste stable et la puissance devient très faible.

Des exemples de décodage par l'algorithme de Viterbi d'un silence et d'une note sont présentées en figures 2.5 et 2.4, avec les descripteurs associés. Elles illustrent les interprétations faites ci-dessus.

## 2.2 Modèle de jeu

En utilisant nos modèles isolés comme briques de base, nous construisons progressivement un modèle réalisant la segmentation/classification. Il ne s'agit pas d'un modèle musicologique, qui intégrerait un style de jeu. On cherche simplement à reconnaitre les éléments musicaux dans un flux continu, sans contrainte stylistique. Pour le tester et le valider, on utilise l'algorithme de Viterbi, l'algorithme classique de décodage de la séquence d'état. Viterbi est un algorithme non-causal. Il utilise la totalité des données pour son calcul, alors qu'un algorithme temps-réel doit se restreindre aux données du passé. Ce décodage offline nous permet de vérifier le bien-fondé de nos hypothèses et de valider chaque étape de la construction du modèle de jeu. La section 2.3 s'intéressera à l'utilisation d'algorithmes temps-réels pour remplacer Viterbi.

### 2.2.1 Classification avec un seul modèle

Le but que l'on se fixe est de construire progressivement un modèle de jeu. La première étape est d'intégrer dans un unique modèle les différentes classes dont les modèles ont été entrainées séparément.

Nous nous proposons donc de fusionner dans un modèle global les deux modèles obtenus auparavant. Ce modèle doit toutefois permettre de reconnaitre la classe d'une séquence d'observation. Auparavant, la vraisemblance était calculée pour les deux modèles, puis elles étaient comparées pour déterminer quel modèle obtenait la valeur maximale. Observons ce qui arrive

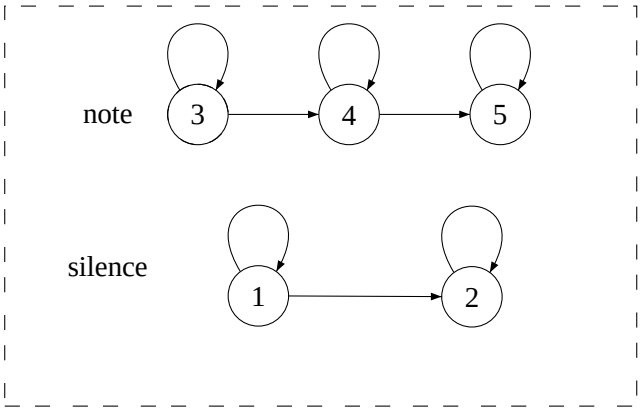


FIG. 2.6: Un modèle global unique pour la classification : il contient les modèles de note et de silence

lorsqu'on intègre les deux modèles de classe — qui sont non-connexes, c'est-à-dire qu'il n'existe pas de transition allant d'un modèle à l'autre — au sein d'un modèle global, comme présenté en figure 2.6, dont les probabilités initiales ($\pi$) sont distribuées équitablement entre les deux états initiaux des modèles de classe. Supposons maintenant qu'on dispose d'une séquence d'observation dont on souhaite retrouver la classe. L'algorithme de Viterbi calculé sur le modèle global va envisager les différents chemins possibles dans ce modèle. Il va envisager des chemins passant par le modèle de note et des chemins passant par le modèle de silence. Aucun chemin passant par les deux modèles à la fois ne peut être envisagé, car aucune transition ne les relie. Au final, l'algorithme doit sélectionner la séquence d'état optimale dans le modèle. Celle-ci correspond au chemin le plus probable. La séquence d'état retenue est composée soit par d'états de la note, soit

d'états du silence, ce qui permet de reconnaitre la classe de la séquence. Le modèle global permet de faire une classification équivalente à celle de la méthode précédente de cette manière.

### 2.2.2 Evaluation des performances de segmentation/classification

Pour évaluer les performances de la segmentation/classification, nous nous basons sur trois critères.

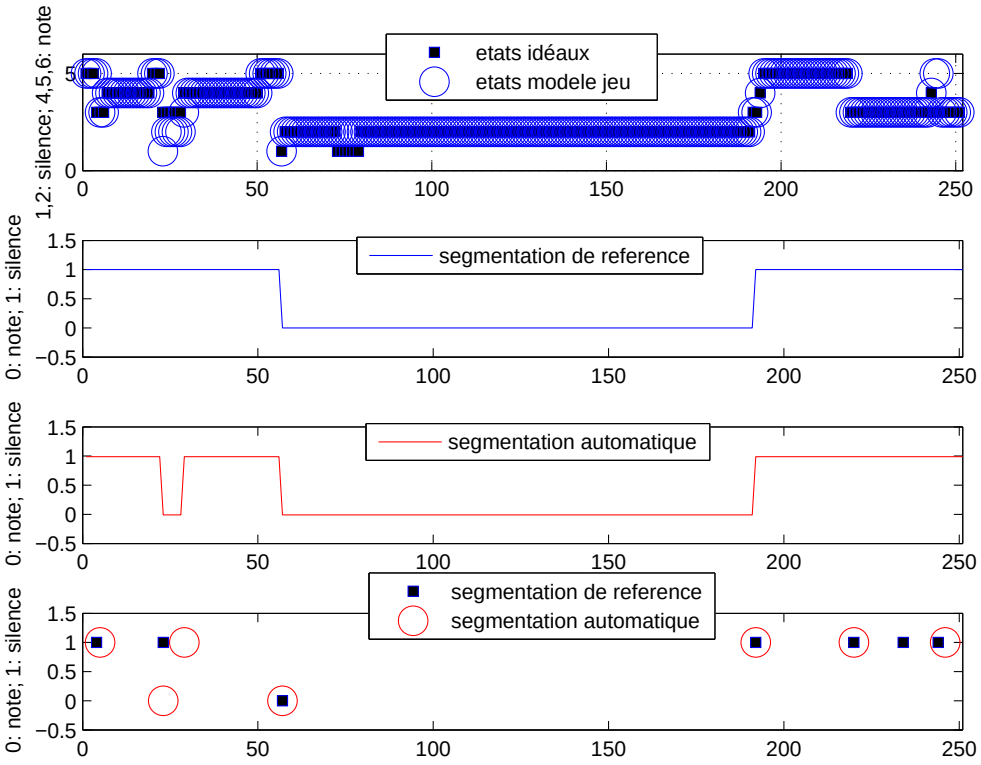


FIG. 2.7: Visualisation des trois critères : Tout en haut, le critère 1 : la séquence idéale (décodée sur les segments isolés) est affichée en carrés noirs, le décodage par le modèle de jeu est affiché avec des cercles. Les deux graphiques du milieu correspondent au critère 2 : un 1 signifie que l'on est dans le modèle de note, un 0 dans un silence. En bas, le critère 3 : les débuts de note et silence sont marqués en noir, les cercles sont les débuts tels que détectés par le modèle de jeu.

#### Critère 1 : Comparaison avec un décodage idéal

Comparaison entre la séquence d'état obtenue sur l'intégralité de l'improvisation et une séquence « idéale ». La séquence idéale est construite de la façon suivante : le décodage est fait sur chaque segment indépendamment, puis les résultats sont concaténés. Le décodage d'un segment isolé est fait grâce au modèle de la classe correspondante. Le nombre d'états différents nous renseigne sur la qualité du modèle de segmentation/classification.

**Critère 2 : Appartenance au bon modèle**

Plutôt que de comparer état par état, on regarde si chaque état décodé appartient au bon modèle, c'est à dire si l'état à l'instant $t$ appartient bien au modèle de la classe jouée à cet instant. Cette mesure est informative, mais dans le cas où plusieurs segments de la même classe sont enchainés, on perd une information sur la qualité de la segmentation. C'est pourquoi on utilise un troisième critère.

**Critère 3 : Pourcentage d'états initiaux bien détectés**

Pour évaluer la justesse de la segmentation, on essaye de faire correspondre chaque état initial décodé avec la segmentation de référence. On tolére un léger décalage : on cherche dans un intervalle de 3 trames avant à 3 trames après, une trame ayant une durée de 11 ms. Les nombres d'états initiaux oubliés et insérés (fausses alertes) sont aussi comptabilisés.

### 2.2.3 Segmentation note-note

Rappelons que notre modèle de jeu ne cherche pas à imposer une grammaire musicale, et qu'il possède un dictionnaire limité d'éléments musicaux. Nous n'envisageons donc pas des techniques complexes pour la segmentation de l'improvisation. Pour commencer l'étude de la segmentation, nous nous inspirons du suivi de partition en concaténant deux modèles de note, afin de reconnaitre et délimiter des séquences de deux notes. Une différence majeure avec le suivi est que les deux modèles de notes sont identiques.

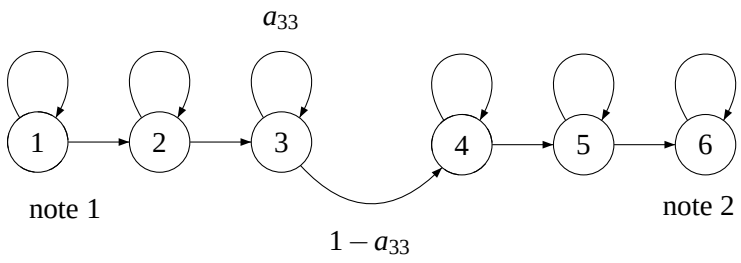


FIG. 2.8: Modèle de deux notes successives. La transition ajoutée est calculée par la formule 2.2

Par définition, la somme des probabilités de transition sortant d'un état doit être égale à 1. D'après la topologie des modèles de classe, la seule transition partant du dernier état est une auto-transition de valeur 1 (voir figures 2.3 et 2.3). Par conséquent, si on ajoute une transition de valeur $a$ vers un autre modèle, l'auto-transition doit être fixée à $1-a$ (voir figure 2.8). La valeur $a$ ne peut pas être choisie arbitrairement ; l'auto-transition traduit la durée de l'état et doit être estimée en fonction de ce critère. En simulant qu'un état est atteint après la fin du décodage, nous estimons la probabilité d'auto-transition $a_{NN}$ de la même manière que dans l'algorithme segmental k-means (voir formule 2.1 page 14). Le calcul permettant d'estimer l'auto-transition du dernier état est :

$$a_{KK} = \frac{||T_{KK}||}{||T_{KK}|| + N_l} \tag{2.2}$$

où $s_K$ est le dernier état et $N_l$ est le nombre d'exemples atteignant $K$.

L'hypothèse que nous formulons est la suivante : la découverte de la séquence d'état optimale nous permet d'identifier les deux notes distinctes. Cette hypothèse est validée suivant le critère

1 : Les statistiques sur la base de test donne un résultat de 10.5% d'états mal reconnus (1291 sur 12291). Une illustration est donnée en figure 2.9.

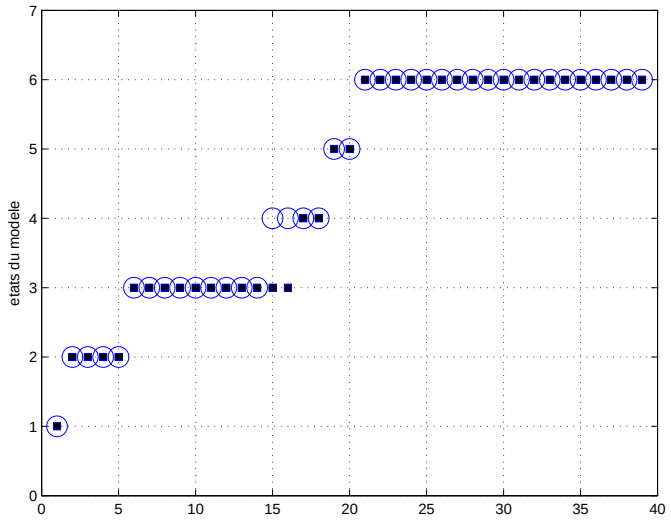


FIG. 2.9: Exemple de décodage grâce au modèle de deux notes successives : on voit que le passage d'un modèle à l'autre n'est pas parfait

Un nouveau modèle est maintenant proposé pour réaliser cette même tâche. Il est constitué d'un seul modèle de note auquel une transition a été ajoutée du début vers la fin (voir figure 2.10). Remarquons que ce n'est pas équivalent au cas précédent, car rien n'impose que seulement deux tours soient faits. Cette fois-ci, 8,0% des états sont mal reconnus (983 sur 12291)

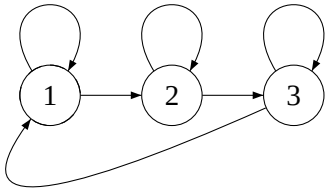


FIG. 2.10: Modèle d'un nombre quelconque de notes enchainées

Nous nous rapprochons de plus en plus d'un modèle de jeu, car comme cela a déjà été remarqué, ce modèle n'impose pas de limite au nombre de notes successives qui peuvent être enchainées. On évalue la qualité du modèle de segmentation pour l'enchainement d'un nombre quelconque de notes. La figure 2.11 donne un exemple sur cinq notes. En concaténant toutes les notes de notre base de test, et en effectuant la recherche de la séquence d'état optimale, 10,6% des états sont mal reconnus (655 sur 6160) suivant le critère 1.

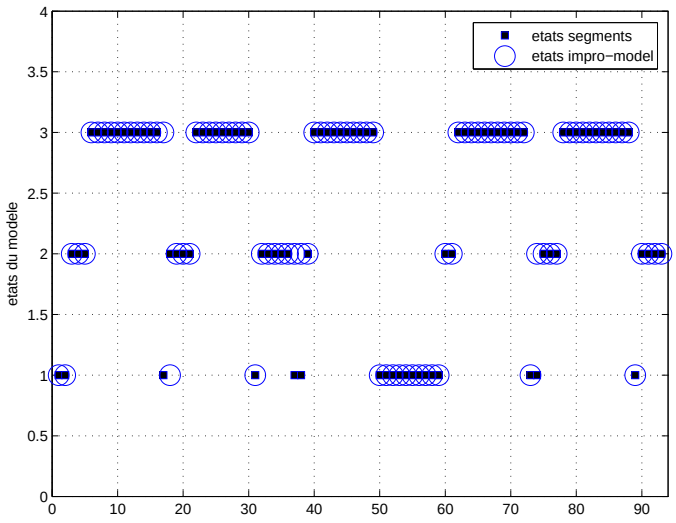


FIG. 2.11: Décodage sur 5 notes, avec le modèle de notes enchainées

### 2.2.4 Segmentation note-silence

Après avoir étudié la segmentation de notes enchaînées, nous nous intéressons à la segmentation note-silence. En reprenant l'approche précédente, un modèle de note et un modèle de silence sont concaténés. Un ensemble de successions note-silence est construit à partir de la base de test, et la pertinence du modèle est évalué selon le critère 1. La seconde étape consiste à relier la fin du silence au début de la note dans le modèle, et à tester la segmentation d'un enchainement note-silence-note-silence-etc. On observe 15.2% d'états mal reconnus (885 sur 5820).

### 2.2.5 Modèle de jeu

Nous sommes prêts à faire un modèle de jeu réaliste. Il contient le modèle de note et le modèle de silence. L'état final du modèle de note est relié aux états initaux de la note et du silence (figure 2.12). La fin du silence est reliée au début de la note. On accepte ainsi des enchainements de notes, entrecoupés d'un silence. Une succession improvisée de notes et de silences est extraite de notre base afin de tester la segmentation. Les résultats selon les trois critères sont les suivants :

- Critère 1 : 17,9% d'états mal reconnus (2076 sur 11569)
- Critère 2 : 7,4% d'états attribués au mauvais modèle (856 sur 11569)
- Critère 3 : 79,23% débuts détectés à 3 trames près (393 sur 496), 20.77% oubliés (103 sur 496), 39.31% insérés (195 sur 496)

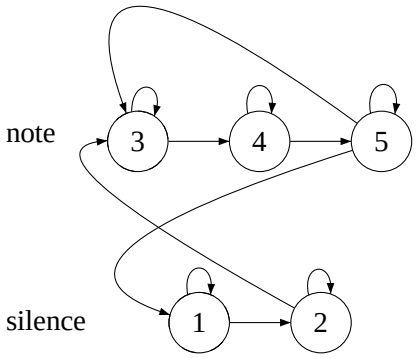


FIG. 2.12: Le modèle de jeu : une note peut être suivie par une note ou un silence, un silence ne peut être suivi que d'une note.

## 2.3 Segmentation et classification en temps-réel

### 2.3.1 Inférence bayésienne en temps-réel

Un article de Andrew J. Davison [Dav03] expose bien la problématique du temps-réel. Les algorithmes offline utilisent la totalité des données disponibles pour calculer un résultat. Dans les systèmes temps-réel, le calcul doit être réalisée de manière séquentielle. Chaque nouvelle observation/donnée permet de mettre à jour les hypothèses que l'on fait sur le système observé. De plus, le calcul effectué sur une nouvelle donnée doit prendre un temps constant, inférieur à l'intervalle entre deux observations. Pour cela il faut pouvoir assurer que le nombre d'opérations est borné. L'estimation séquentielle est une série pondérée de combinaisons des anciennes et des nouvelles données. Une technique qui peut être adoptée pour réduire le nombre d'hypothèses est de "prédire le futur". Ainsi certaines éventualités peuvent être rejetées, plutôt qu'envisagées.

L'inférence bayésienne est une manière d'utiliser les méthodes probabilistes dans laquelle la probabilité associée à un ensemble d'observations est utilisée pour associer un degré de croyance en une hypothèse. Une probabilité à priori à toutes les possibilités observables avant de recevoir une donnée. $P(X)$ Dès qu'on reçoit une donnée $Z$, on l'incorpore en calculant la probabilité à postériori $P(X|Z)$ grâce au théorème de Bayes. Elle représente notre connaissance du système.

### 2.3.2 Première proposition : Adaptation de Viterbi

Par segmentation en temps-réel, on entend isoler les unités musicales peu après qu'elles aient été émises. Le délai peut être un paramètre de l'algorithme.

Une première idée qui vient à l'esprit pour réaliser l'inférence de manière causale est d'utiliser la variable $\alpha$ de la procédure forward-backward (voir 1.4). En effet, $\alpha_t$ nous renseigne sur l'état le plus vraisemblable en sommant les probabilités de tous les chemins possibles. $\alpha_t$ est calculé grâce à $\alpha_{t-1}$ et $O_t$. La variable $\delta$ de Viterbi donne le même type de renseignement mais en utilisant seulement les chemins les plus probables pour calculer la vraisemblance des états. Toutefois, la séquence constituée par l'état le plus probable pour chaque instant n'est pas nécessairement une séquence valide, car elle peut représenter un parcours impossible dans le modèle, ie. qui passe d'un état à un autre sans qu'il existe de transition. La variable $\psi$ de l'algorithme de Viterbi mémorise pour chaque état et à chaque instant l'état précédent le plus probable, ce qui permet d'identifier la

séquence optimale en la parcourant en arrière à la fin des observations (backtracking).

Il est envisageable de mettre à jour les variables $\delta$ et $\psi$, et d'effectuer un backtracking à chaque instant t. L'opération de backtracking est tellement trivale, qu'on ne peut pas dire qu'elle représente un coût de calcul à maîtriser dans un cadre temps-réel. En effet, la deuxième contrainte des algorithmes en temps-réel – outre qu'ils soient causaux – est d'avoir un nombre d'opérations bornées.

A chaque instant, on a le chemin le plus probable, cependant il est possible qu'à l'instant suivant, un chemin complètement différent soit le plus vraisemblable. Il faut choisir un chemin à un moment donné afin de procéder à la segmentation.

### 2.3.3 Filtrage particulaire

#### Présentation

Pour la vision des systèmes robotiques, on est confronté au problème du tracking : identifier en permanence la localisation d'un objet dans l'espace. Des méthodes simples où le modèle est initialisé à chaque instant par la configuration du modèle à l'instant précédent peuvent être induit en erreur par des maxima locaux. (détails, hypothèses moins vraisemblables éliminées alors qu'elles pourraient être prouvées plus tard). Il est préférable d'utiliser la connaissance du passé pour prédire, avec un degré d'incertitude, l'endroit où un objet sera au temps suivant, et de réduire la recherche à cet endroit.

Le filtrage particulaire [AMGC02] permet de représenter la probabilité à posteriori comme un ensemble fini de particules pondérées, qui approximent une densité de probabilité de forme quelconque. Le tracking grâce au filtrage particulaire se déroule en 3 étapes :

- resampling, où les particules sont transformées en un ensemble de particules de poids égaux, dont la concentration est en accord avec la densité de probabilité.
- dispersion des particules en fonction d'un modèle de mouvement.
- observation et réévaluation des poids des particules en fonction de la nouvelle donnée.

Le filtrage particulaire permet de propager plusieurs hypothèses, et de ne pas écarter les hypothèses moins probables, qui pourront être vérifiées plus tard.

#### Filtrage particulaire pour les modèles de Markov

On peut utiliser le filtrage particulaire pour évaluer la position dans un modèle de Markov à un instant donné. Le filtrage particulaire appliqué aux modèles de Markov est utilisé dans le suivi de partition de l'Ircam et brièvement évoquée dans [Con06].

Voici l'algorithme du filtrage particulaire tel que nous l'avons implanté. Il met en concurrence $M$ particules distribuées sur les états du modèle.

- Initialisation : Les $M$ particules sont distribuées grâce à un tirage suivant la distribution de probabilité initiale ($\pi$).
- Boucle infinie :

  1. Pondération : Chaque particule $p_m$ est associée à un poids $w_m$. Ce poids est égal à la probabilité de l'observation $O_t$ pour l'état $s_i$ dans lequel elle se trouve : $w(s_i, O_t) = b_i(O_t)$. Les poids sont normalisés tels que $\sum_{m=1}^{M} p_m = 1$. La distribution des poids sur les états forme une approximation de la densité de probabilité de la position dans le modèle.

  2. Rééchantillonnage : Les particules les plus lourdes donnent naissance à de nouvelles particules, tandis que des particules de poids trop faible disparaissent. Plus formellement, on effectue une redistribution des particules dans le modèles en utilisant la densité calculée à l'étape précédente.

3. Propagation (ou prédiction) Chaque particule est propagée suivant la distribution de transition de son état : $A_i = \{a_{ij}\}\ \ j = 1 \ldots N$. Les transitions sont choisies par tirage.

La phase de réechantillonnage permet d'éliminer les trajectoires dont les poids sont faibles et multiplier les trajectoires dont les poids sont forts. Les particules sont ainsi concentrées dans les zones pertinentes. Sans cette phase se produit le phénomène de dégénerescence, dans lequel toutes les particules sauf une possèdent un poids négligeable.

### 2.3.4 Résultats

Pour évaluer la technique du filtrage particulaire, nous avons repris le modèle de classification présenté en 2.2.1, et observé les trajectoires des particules. Tel que présenté ci-dessus, et dans notre cas d'application, le filtrage particulaire ne donne pas de bons résultats. Nous avons en effet observé que l'algorithme était induit en erreur par des maximas locaux. Prenons l'exemple présenté en figure 2.13, qui utilise 50 particules pour décoder la séquence d'état d'une note (état 3, 4 et 5 ; cf. modèle en figure 2.6). Lors de l'initialisation, les 50 particules sont réparties en fonction de $\pi$ sur les états 1 et 3. Les particules commencent à se propager en t=2. Comme on le voit sur la figure, la probabilité de $O_3$ est la plus forte pour l'état 2, et la phase de réechantillonnage élimine les particules des autres états. Les 50 particules restent ensuite bloqués sur le deuxième état, en accord avec la topologie du modèle. Pourtant, la suite des observations aurait largement favorisé un chemin passant par le modèle de note, si l'hypothèse avait été envisagée plus longtemps.

Nous avançons deux hypothèses sur l'échec de cette première implantation :

– Une mauvaise maîtrise de la phase de rééchantillonnage. L'algorithme que nous avons utilisé est le rééchantillonnage résiduel (residual resampling). Une meilleure compréhension de ces techniques permettrait éventuellement d'identifier la cause de l'échec.
– Une mauvaise adéquation du modèle avec les observations. En effet, les modèles ont été entrainés pour reconnaitre des observations très variées et ne sont pas toujours en accord avec la totalité des observations.

Quoi qu'il en soit, un rééchantillonnage à chaque nouvelle observation fait tomber les particules dans des maximas locaux. C'est pourquoi nous avons effectué le rééchantillonnage à des intervalles réguliers. La figure 2.14 présente le même exemple, avec un rééchantillonnage toutes les sept observations. Cependant, le problème réapparait lorsque les maximas locaux sont éloignés de plus de sept observations.

#### Condition sur le réechantillonnage

La littérature propose une condition sur le rééchantillonnage lorsque la technique précédente échoue. Le nombre $\hat{N}_{\text{eff}} = \frac{1}{\sum_{m=1}^{M}(w_m)^2}$ représente le nombre de particules effectives et est une mesure de la dégénerescence[2]. $\hat{N}_{\text{eff}}$ est maximisé lorsque tous les poids sont égaux, et minimisé lorsqu'un seul poids est non-nul. Quand $\hat{N}_{\text{eff}}$ est inférieur à un seuil, on procède au rééchantillonnage. Toutefois, il faut trouver une valeur optimale pour le seuil.

#### Evaluation des résultats

Ces mesures ont été réalisées avec 50 particules et un rééchantillonnage résiduel toutes les 7 observations. Les résultats de la classification sont satisfaisant. Les notes sont classées correctement à 97,1% et les silences à 91.

– Sur le modele de 2 notes enchainées, 43,2% des états mal sont reconnus.

[2]toutes les particules sauf une possèdent un poids négligeable

)

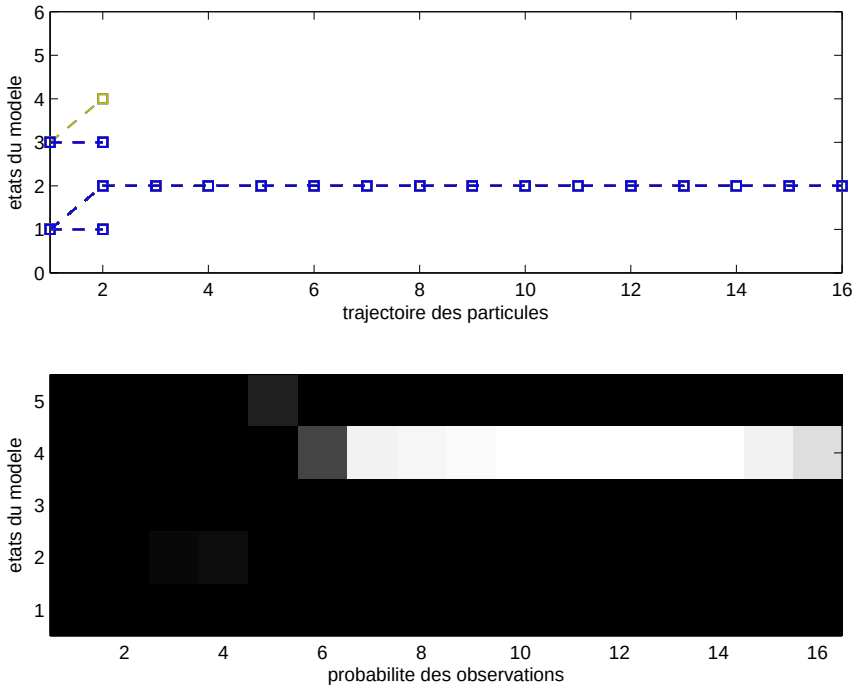


FIG. 2.13: Filtrage particulaire, maximum local (a) : En haut, les trajectoires des particules. En bas, les probabilités des observations $b_j(O_t)$ sont représentés (le blanc représente une probabilité forte). Avec un rééchantillonnage après chaque observation, le maximum local induit les particules en erreur

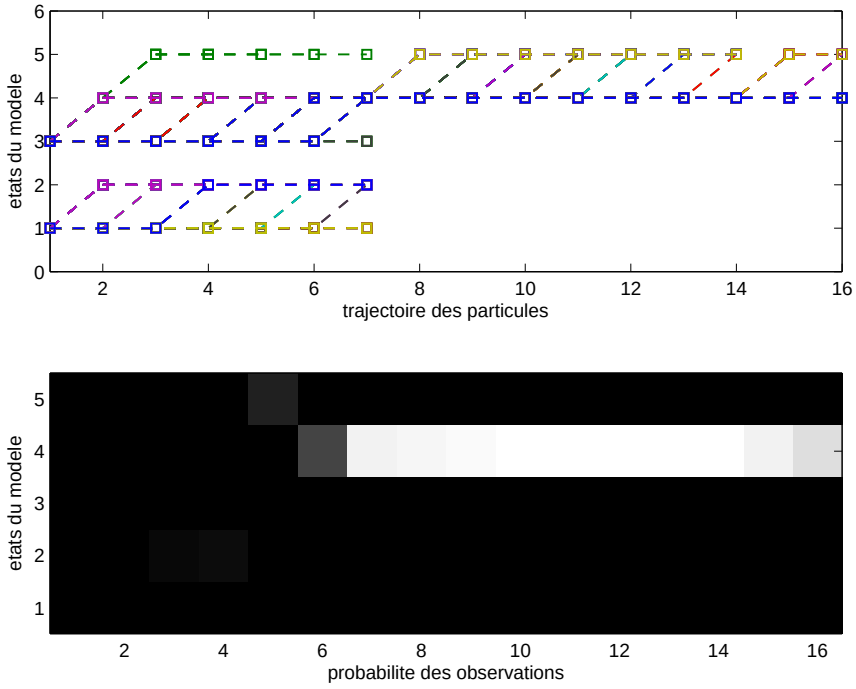


FIG. 2.14: Filtrage particulaire, maximum local (b). Avec un rééchantillonnage toutes les sept observations, plus d'hypothèses sont envisagées et le maximum local est évité.

- Le décodage de 345 séquences de 2 notes avec le modèle d'un nombre quelconque de notes enchainées a obtenu 35,5% d'états mal reconnus.
- Le décodage d'une séquence de 346 notes sur le modèle d'un nombre quelconque de notes enchainées a donné 42.6% d'états mal reconnus.
- Le décodage d'un séquence de 110 événements a donné 31,7% d'états mal reconnus.

Les trois critères sur le modèle de jeu ont donné les résultats suivants :

1. 38,4% d'états mal reconnus
2. 9,7% d'états attribués au mauvais modèle (1125 sur 11569)
3. 39,72% débuts détectés à 3 trames près (197 sur 496)
4. 60,28% oubliés (299 sur 496)
5. 49,40% insérés (245 sur 496)

Une analyse de ces résultats reste à faire.

# Conclusions et Perspectives

Durant ce stage, nous avons modélisé des événements musicaux en utilisant uniquement les descripteurs de Yin. Ces modèles ont ensuite été utilisés pour la segmentation et classication de l'improvisation. Enfin le décodage/identification a été adaptée pour être calculable en temps-réel.

Plus de modes de jeu peuvent être modélisés. Pour le glissando, on peut imaginer reprendre la topologie du modèle de note pour l'entrainer. La dérivée de la hauteur pendant le second état ne sera alors plus nulle. Pour le vibrato, un modèle nouveau peut-être créé, acceptant la variation alternativement positive et négative de la hauteur.

La nécessité de mieux coller aux observations a été mise en évidence. Un travail actuellement en cours est l'utilisation de mélanges de gaussiennes pour représenter les densités $b_j$ des modèles. La durée d'un état peut être mieux modélisée. Pour avoir une distribution dont le mode n'est pas en 1, on peut lier $n$ états partageant la même distribution de probabilité. La distribution résultante est une distribution géométrique dont le mode est éloigné de 1. En les combinant en parallèle, on peut avoir des distributions multimodales.

Pour le filtrage particulaire appliqué à la segmentation en temps-réel, on sait qu'au fur et à mesure que les particules disparaissent, leur trajectoires sont éliminées. Si l'on regarde assez loin en arrière dans le temps des observations, on observe qu'au delà un certain point, les particules du présent partagent la même histoire. Dès lors, on sait que d'autres hypothèses ne seront plus envisagées, et on peut considérer le décodage comme accepté. La segmentation/classification peut alors avoir lieu.

# Bibliographie


[AMGC02] S. Arulampalam, S. Maskell, N. Gordon, and T. Clapp. A tutorial on particle filters for on-line non-linear/non-gaussian bayesian tracking, 2002.

[Con04] Arshia Cont. Improvement of observation modeling for score following. Master's thesis, Université Pierre et Marie Curie, PARIS VI, 30 June 2004.

[Con06] Arshia Cont. Realtime audio to score alignment for polyphonic music instruments using sparse non-negative constraints and hierarchical hmms. In *IEEE International Conference in Acoustics and Speech Signal Processing (ICASSP)*. Toulouse, May 2006.

[Dav03] Andrew J. Davison. Modelling the world in real time : ow robots engineer information. *Phil. Trans. R. Soc. Lond. A*, 2003.

[dCK02] Alain de Cheveigné and Hideki Kawahara. YIN, a fundamental frequency estimator for speech and music. *JASA*, April 2002.

[JR90] B. H. Juang and L. R. Rabiner. The segmental k-means algorithm for estimating the parameters of hidden markov models. *IEEE Trans. Accoust., Speech, Signal Processing*, ASSP-38(9) :1639–1641, September 1990.

[OVWY94] J. J. Odell, V. Valtchev, P. C. Woodland, and S. J. Young. A one pass decoder design for large vocabulary recognition. In *HLT '94 : Proceedings of the workshop on Human Language Technology*, pages 405–410, Morristown, NJ, USA, 1994. Association for Computational Linguistics.

[Pau92] Douglas B. Paul. An efficient a* stack decoder algorithm for continuous speech recognition with a stochastic language model. In *HLT '91 : Proceedings of the workshop on Speech and Natural Language*, pages 405–409, Morristown, NJ, USA, 1992. Association for Computational Linguistics.

[Rab90] Lawrence R. Rabiner. A tutorial on hidden markov models and selected applications in speech recognition. pages 267–296, 1990.

[Rap98] Christopher Raphael. Automatic segmentation of acoustic musical signals using hidden markov models. 18 January 1998.

[Sch04] Diemo Schwarz. *DATA-DRIVEN CONCATENATIVE SOUND SYNTHESIS*. PhD thesis, Université Paris 6 - Pierre et Marie Curie, 23 January 2004.