REAL-TIME IMPLEMENTATION OF HMM-BASED CHORD ESTIMATION
IN MUSICAL AUDIO

Taemin Cho, Juan P. Bello
Music and Audio Research Laboratory (MARL)
New York University, New York, USA
tmc323@nyu.edu

ABSTRACT

In this paper, we implement a real-time chord estimation system based on the hidden Markov model (HMM) using chroma features. HMMs, with the Viterbi decoding algorithm, have proven a powerful tool for chord recognition in polyphonic music signals. However, the direct application of the traditional, non-causal, decoding approach is not the optimal choice for real-time processing, with limited memory capabilities and no access to future observations. We propose a system of buffers and a modified decoding process that approximates offline results while minimizing the system’s latency.

1. INTRODUCTION

The automatic recognition of musical attributes such as melody, rhythm, and harmony from polyphonic music signals is an important area of development in computer music and machine listening research. Particularly, the characterization of the harmonic content (e.g. chords and keys) finds applicability in tasks such as song thumbnails, automatic music transcription and the development of interactive music systems. Some of these tasks require attribute recognition to occur in real-time, allowing performers to effectively communicate with computers using the sound of their instruments without recourse to MIDI devices or triggering operations.

Unfortunately, most existing state of the art approaches to chord recognition are designed for offline processing [1, 4, 6], thus precluding their usefulness to realtime applications. There are a few existing realtime works; however, they either do not use adaptive machine learning [3] or have not yet been implemented as realtime systems [7].

In this paper, we propose a method for real-time chord recognition in polyphonic audio signals. We adapt the hidden Markov model (HMM) using chroma vectors formulation to a real-time context by introducing modifications to the decoding process intended to approximate offline results with minimal decoding latency.

2. BACKGROUND

Pitch class profiles (PCP), commonly known as chroma features, represent the distribution of a signal’s frequency content across the 12 semitones of the chromatic scale, making them the features of choice for analysis tasks such as chord and key estimation. Fujishima [3] pioneered the use of chroma features for audio-based chord recognition. His system, intended for real-time operation, uses a simple pattern matching approach that performs well on samples containing a single instrument, but does not cope well with the complexities of polyphonic and multi-instrumental music.

In an attempt to minimize the shortcomings of the simple pattern matching approach, Sheh and Ellis [6] introduced the Chord estimation from the chroma features using HMMs. Although their results were poor (maximum accuracy of 26.4%), their work laid a foundation for future studies. The combination of chroma features and hidden Markov models is now the de facto standard for audio-based chord recognition. All approaches presented at the MIREX 2008’s audio chord detection task are fundamentally variations of this basic architecture. For a comparative review see [4].

2.1. Approach

Our approach is a re-implementation of the system discussed in [1], one of the top performers in MIREX 2008\(^1\). It begins with an efficient algorithm for the calculation of the constant $Q$ transform $X_{cq}$ as the multiplication of the short-time Fourier transform and the complex conjugate of a pre-calculated sparse kernel [2]. $X_{cq}$ is then used to compute the chroma features as $O_b = \sum_{m=0}^{M-1} |X_{cq}(b + m\beta)|$, where $\beta$ is the number of bins per octave, $b = 1 \cdots \beta$ are the chroma bin indices, and $M$ is the total number of octaves from a minimum frequency $f_{\text{min}}$. In our implementation, $\beta = 36$, $M = 6$ and $f_{\text{min}} = 130.81\,\text{Hz}$ are used. Finally the chroma feature vectors are quantized into 12 bins using a Gaussian filter bank.

A hidden Markov model, $\lambda$, is defined by the initial state distribution $\pi$, the set of state to observation probability dis-
3. REALTIME DECODING SYSTEM

Unlike an offline system, an online system does not have access to the entire signal at the onset, nor does it know the length of the signal. Data is fed to the system in real time and must be processed with the lowest latency possible.

To accommodate these constraints, we use two different types of buffers, an audio input buffer and an observation buffer. The audio input buffer captures block size of audio samples from incoming audio signals. Chroma features are extracted from this buffer each time new hop size of samples comes into the buffer. The observation buffer keeps L number of the extracted chroma vectors for local decoding processes. Figure 1 shows the process at times t and t + 1.

3.1. The Viterbi Algorithm

The Viterbi Algorithm [5] is the most commonly used decoding algorithm for HMMs. For a given model λ and a T-long observation sequence O = {O₁, O₂, ..., O₇}, it finds the single best state path Q = {q₁, q₂, ..., q₇} that maximizes P[O|Q, λ], which is the same as maximizing P[Q, O|λ]. The highest probability, δᵢ(1), along a single path ending in state Sᵢ, from the beginning of the observation sequence to time t, is defined as:

\[ \delta_i(1) = \max_{q_1, q_2, ..., q_{t-1}} P[q_1 q_2 \cdots q_t = S_i, O_1 O_2 \cdots O_t | \lambda] \]  

(1)

By induction, δᵢ(ₗ), can be denoted as:

\[ \delta_i(\lambda) = \max_{q_1, q_2, \cdots, q_{\lambda-1}} \delta_{i-1}(i) a_{ij} b_j(O_t), \quad t > 1 \]  

(2)

With an array \( \psi(\lambda) \), a container for the argument which maximizes (2) at time \( t \) and state \( S_i \), the complete Viterbi procedure for the given N-state model \( \lambda = \{A, B, \pi\} \) and the observation sequence \( O = \{O_1, O_2, \cdots, O_T\} \) can be stated as follows:

1. Initialization:

\[ \delta_i(1) = \pi_i b_i(O_1), \quad 1 \leq i \leq N \]  

(3a)

\[ \psi_i(1) = 0, \quad 1 \leq i \leq N \]  

(3b)

2. Recursion:

\[ \delta_i(t) = \max_{1 \leq i \leq N} \{\delta_{i-1}(i) a_{ij} b_j(O_t), \quad 2 \leq t \leq T, \quad 1 \leq j \leq N \} \]  

(4a)

\[ \psi_i(t) = \arg \max_{1 \leq i \leq N} \{\delta_{i-1}(i) a_{ij}, \quad 2 \leq t \leq T, \quad 1 \leq j \leq N \} \]  

(4b)

3. Termination:

\[ P^* = \max_{1 \leq i \leq N} \{\delta_T(i)\} \]  

(5a)

\[ q_T^* = \arg \max_{1 \leq i \leq N} \{\delta_T(i)\} \]  

(5b)

4. Backtracking:

\[ q_t^* = \psi_{t+1}(q_{t+1}^*), \quad t = T - 1, T - 2, \cdots, 1 \]  

(6)

The obtained sequence \( Q^* = \{q_1^*, q_2^*, \cdots, q_T^*\} \) from (5b) and (6) is the single best state path, which has the maximum likelihood, \( P^* \).

3.2. A Variation on the Viterbi Algorithm

The standard Viterbi algorithm shown in Section 3.1 is designed for decoding on an independent observation sequence. Therefore, the direct adaptation of it to the observation buffer isolates the current decoding process from previous processes. This computational independence is critical because the effectiveness of dynamic programming of the Viterbi algorithm is based on previous computations. In order to include previous calculations without losing dynamic programming properties, we propose a variation of the Viterbi algorithm.
The basic idea is to substitute the fixed $\pi$-based initialization step (3) of the standard algorithm with an initialization that reuses previous buffer calculations. When the total number of extracted observations $T = L$, then the observation buffer is filled, and the decoding process is triggered (see Figure 1). The standard Viterbi algorithm is only used at this time to initialize $\delta_t(i)$ and $\psi_t(i)$, and calculate $\delta_t(i)$ and $\psi_t(i)$ at $t = 2 \cdots L$, thus generating the first decoded sequence, $Q_L$. At $T > L$ and $t = T$, the modified Viterbi algorithm is used to decode the observation sequence. As we already have $\delta_t(i)$ and $\psi_t(i)$ for $t \leq T - 1, T - L + 1$, we can skip the initialization step (3) and most calculations at the recursion step (4). The resulting variation is denoted as:

1. Recursion:
   \[
   \delta_T(j) = \max_{1 \leq i \leq N} [\delta_{T-1}(i) a_{ij}] b_j(O_T), \quad T > L, \quad 1 \leq j \leq N
   \]
   \[
   \psi_T(j) = \arg\max_{1 \leq i \leq N} [\delta_{T-1}(i) a_{ij}], \quad T > L, \quad 1 \leq j \leq N
   \]

2. Termination:
   \[
   q_T^* = \arg\max_{1 \leq i \leq N} [\delta_1(i)]
   \]

3. Backtracking:
   \[
   q_t^* = \psi_{t+1}(q_{t+1}), \quad t = T - 1, T - 2 \cdots, T - L + 1
   \]

By using the results of previous calculations we make dynamic programming possible in real-time, operating very much in the same way as the offline process, with the exception of the partial backtracking process (9). Therefore, we expect the results of the modification to be closer to those of non-realtime decoding. In addition, as the modified algorithm skips many duplicated procedures, we expect it to be faster than the standard algorithm.

Figure 2. The decoded sequence matrix with $L = 5$, $t = 1, 2, \cdots, 10$. The decoded sequences are stacked into a $L \times L$ matrix top to bottom. At time $t = 10$, $Q_{10}$ is pushed into the matrix and $Q_5$ is pushed out from it. The dotted square shows the matrix at time $t = 9$.

3.3. Frame-Level Chord Estimation

As seen in Figure 2, after $2 \times L - 1$ frames, the decoded sequences always form an $L \times L$ lower triangular matrix. The estimated chord at time $t$ is the mode of the first column vector of the matrix. For example, in Figure 2, the final estimated chords at times $t = 5$ and 6 are both F.

3.4. Decoding Latency

The decoding latency is determined by the block size, the hop size and by the length of the observation sequence - $L$. Due to the fact that the recognized chord is the chord of the $L$th frame and assuming that the chord represents the block size of the signal, the latency can be calculated as:

\[
\text{latency} = \frac{(\text{block size}/2 + \text{hop size} \cdot (L - 1))}{\text{Sample rate}} \text{ (sec.)}
\]

Both the accuracy and the latency are highly dependent on $L$. We argue that the modified Viterbi algorithm is capable of simultaneously increasing accuracy and decreasing the latency.

4. IMPLEMENTATION AND EXPERIMENTS

Our system, shown in Figure 3, was developed on an Apple MacBook Pro (2.33 GHz Intel Core 2 Duo, 2 GB 667 MHz DDR2 SDRAM). The prime consideration for this implementation was having every DSP calculation finished within the time interval determined by the hop size. For maximum performance, we used a C-based library optimized for SSE\(^2\) - the vDSP Accelerate framework. Coding was done in C++ and wrapped by Objective-C++ 2.0 within the Cocoa framework. In our experiments, it takes an average of 0.35\( ms\)

\(^2\)Intel’s Streaming SIMD Extensions (called SSE) is a 128-bit SIMD vector extension to the x86 ISA
to generate a chroma vector, while the decoding times for 30-long observation sequences are 3.8 and 1.2 ms using the standard and modified Viterbi algorithms, respectively.

For the experiments, we performed 12-fold cross validation on a set of 169 annotated Beatles recordings, the same used for the 2008 MIREX evaluation. Each song in the set is a CD quality wav file (2 channels, 44100 Hz and 16 bit PCM). We use 8192 and 2048 samples for the block size and hop size respectively, and use a 44100 Hz sampling rate. Model training was unsupervised and performed offline.

For the evaluation of the modified Viterbi algorithm, the realtime decoding process was computed with 6 different observation sequence lengths, \( L = 5, 10, 15, 20, 25 \) and 30. We compared with non-realtime results and the results of the standard Viterbi algorithm in real-time.

We accomplish the initial goal of this paper, namely, to make the results of realtime decoding as close as possible to the results of non-realtime decoding while minimizing latency. However, the inevitable and comparatively long decoding latency still remains a problem, made worse by the proportionality between this latency and the accuracy of the system. Simple solutions, such as a reduction of the block size, result in unstable and noisier features that negatively affect accuracy.

The experimental results clearly show that the real-time system can only be as good as the offline system is. Thus, future work will be focused on improving both the feature extraction and the modeling stages, by exploring means beyond the using chroma features with the HMM.

6. ACKNOWLEDGMENTS

This work is made possible by grant LG-06-08-0073-08 from the U.S. Institute of Museum and Library Services.

7. REFERENCES


