On the Relative Importance of Individual Components of Chord Recognition Systems

Taemin Cho and Juan P. Bello, *Member, IEEE*

***Abstract*—Most chord recognition systems share a common architecture comprising two main stages: feature extraction and pattern matching, and two optional sub stages: pre-filtering and post-filtering. Understanding the interaction between these basic components is very important not only for achieving optimal performance, but also for assessing the potential and limitations of the system. Unfortunately, there are no studies that sufficiently evaluate the effects of the different approaches to each processing step and the interactions between these steps. In this paper we attempt to remedy this deficiency by performing a systematic evaluation encompassing a wide variety of techniques used for each processing step. In our study we find that filtering has a significant impact on performance, but providing musical context information in the transition matrix is rendered moot by the need to enforce continuity in the estimations. We discovered that the benefits of using complex chord models can be largely offset by an appropriate choice of features. In addition, the initial performance gap between different features were not fully compensated by any subsequent processing stages.**

***Index Terms*—Automatic chord recognition, chroma, Gaussian mixture models (GMMs), hidden Markov models (hmms).**

I. Introduction

A MUSICAL chord is defined as a group of notes sounding either simultaneously or in close succession. Sequences of these events contain important information not only about the pitch and harmonic content but also rhythm and other structural cues. Furthermore, chord transcriptions can capture many important aspects of music in compact form, and have therefore attracted attention as a convenient and useful resource for a wide variety of music applications such as music segmentation [1], cover song identification [2]–[4] and mood classification [5]. However, chord transcriptions are not readily available for most recorded music and can only be generated by highly-trained musicians with much time and effort. Accordingly, automated methods to identify chords from recorded music have been widely studied, and a number of techniques have been proposed in the literature.

The basic chord recognition procedure is composed of two distinct steps: feature extraction and pattern matching. In the first step, a given audio signal is cut into short frames, and each frame is transformed into an appropriate harmonic feature vector. In the second step, chord labels are determined by measuring the fit between the harmonic features and a set of predefined chord models. In order to compensate for the errors that may have occurred during these steps, additional filtering steps are commonly employed after the first step (pre-filtering), after the second step (post-filtering), or both. Arguably, most existing chord recognition systems follow this underlying framework, only with different approaches and parameters for each of these steps. Indeed, the many contributions described in the literature focus mainly on alternative algorithms for one or more steps of this framework, resulting in many different feature extraction methods, a variety of types of chord models, and various filtering approaches.

The performance of a chord recognition system is determined by a complex interplay between the techniques used in each processing step of the system. Hence, understanding the effects of these techniques, their interrelationship and relative importance in recognition results is very important not only for achieving optimal performance, but also for assessing the potential and limitations of the system. However, chord recognition systems are often developed independently, with different combinations of techniques for each processing step. Furthermore, in many cases, they are evaluated with different chord class sets and data sets. Thus, it is often hard to see which technique has the greatest impact on recognition performance, or how each technique would potentially affect systems other than the one in which it was developed. Although the MIREX[1] evaluation provides some insight into the effects of different chord recognition systems, a direct comparison of their performance does not say much about which components of the systems actually impact performance. Unfortunately, only a few studies, including our own [6], have investigated the effects of the different approaches to each processing step and the interactions between them. However, these studies are still limited to a particular system, e.g., a typical HMM-based system [7], or a particular stage, e.g., the feature extraction stage [8], [9], the filtering stage [10].

In this paper we attempt to remedy this deficiency by examining a wide variety of feature extraction methods, chord

Manuscript received June 21, 2013; revised September 12, 2013; accepted December 09, 2013. Date of publication December 23, 2013; date of current version January 10, 2014. This work was supported by the National Science Foundation under Grant IIS-0844654 and by the U.S. Institute of Museum and Library Services under Grant LG-06-08-0073-08. The associate editor coordinating the review of this manuscript and approving it for publication was Dr. Emmanuel Vincent.

The authors are with the Music and Audio Research Laboratory (MARL), New York University, New York, NY 10012 USA (e-mail: tmc323@nyu.edu; jpbello@nyu.edu).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TASLP.2013.2295926

[1]Music Information Retrieval Evaluation eXchange : an annual evaluation campaign for Music Information Retrieval algorithms. http://www.music-ir.org/mirex/wiki/MIREX_HOME

models, and pre- and post-filtering approaches, thereby providing valuable information that can guide future developments in automatic chord recognition. This work is an extension of our previous work [6], which concentrated on variations of filtering parameters and the testing of different pattern matching approaches but did not investigate feature extraction approaches. We expand by including various types of features and a popular pre-filtering technique known as beat-synchronization. In addition, we use a database consisting of 495 chord annotated songs, more than double the size of the database used in the previous study. We also provide a more in-depth analysis of the relative impact of system components.

It must be clarified that covering the full range of possible approaches is beyond the scope of this study. Instead, we choose to use several of the most common and representative techniques per processing step, each of which can be found in state-of-the-art chord recognition systems. Finally, all evaluations in this paper are performed on the detection of major, minor and no chords, the chord set currently used in MIREX evaluations. The remainder of this paper is organized as follows: Section II introduces the fundamental architecture of chord recognition systems with a brief outline of each stage in this architecture; Section III to Section VI describe the details of common techniques used in each stage; Section VII presents the experimental setup and evaluation methodology; In Section VIII we discuss the results; while Section IX includes our conclusions and directions for future work.

II. Architecture of a Chord Recognition System

As mentioned in the introduction, most chord recognition systems share a common architecture comprising two main stages: feature extraction and pattern matching, and two optional sub stages: pre-filtering and post-filtering, as illustrated in Fig. 1. The following subsections provide a brief overview of each stage in this architecture.

A. Feature Extraction

The role of this stage is, as its name implies, to extract appropriate harmonic features from the audio signal. For over a decade since Fujishima [11], chroma features have been the most popular features used for chord recognition. A chroma feature vector, also referred to as pitch class profile (PCP), represents the energy distribution of a signal's frequency content across the 12 pitch classes of the equal-tempered scale. A temporal sequence of these chroma vectors is often called a chromagram [12]. Because a chord is composed of a set of notes, and it is assumed that the chord label can be determined by the occurrence of those notes regardless of octave, chroma features remain the de facto standard signal representation for chord recognition systems. Thus, the feature extraction techniques that have been developed are mostly methodological variations or refinements of the original approach.

In our experiments, we investigate the impact of common variants of chroma features on chord recognition performance, and explore the implications of these variants for the subsequent processing stages. The implementation details of chroma features and their variants are discussed in Section III.

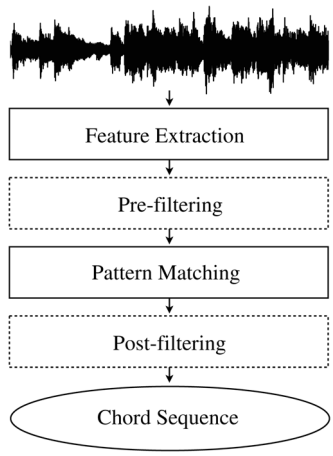

Fig. 1. Fundamental Architecture of a chord recognition system.

B. Pattern Matching

Because of the prevalence of chroma features in chord recognition systems, what makes each system unique is largely dependent on its decision mechanism. Although few studies use discriminative machine learning techniques such as Conditional Random Field [13] and structured Support Vector Machine [14], the majority use generative chord models such as Gaussian models [1], [15]–[17]. Generative chord models primarily focus on describing the distribution of features for individual chords, while the discriminative models focus on determining the classification boundaries between different chords in feature space. Since we only consider "common" approaches in this study, we discard discriminative models.

In the case of generative approaches, the chord label of each frame is determined by finding the chord model that best fits that frame. Thus, in order to identify N different chords, N distinct chord models are necessary. These chord models can be classified into two major categories in terms of how they are constructed. One is generated manually using musical knowledge, and the other is derived stochastically from examples of real-world music. In the former case, the most popular approach is to use hand-crafted chord templates based on the classical approach proposed in 1999 by [11], often used due to its simplicity. In the latter case, Gaussian-based stochastic chord models have become popular with the increasing availability of hand-labeled data. In this paper, we investigate the effects of increasing the complexity of these models, and discuss its relative importance compared to parameters in other processing stages. The specific methods used to generate the models are described in Section IV.

C. Filtering

In chord transcription, determining chord boundaries is as important as identifying chord labels. Accordingly, the frame rate of a chromagram is chosen to be significantly faster than the typical rate of chord changes in music.[2] However, in this case, chroma features become susceptible to undesirable local variability such as transients and noise in the signal. The optional

[2]The average rate of chord change in our dataset (consisting of 495 popular songs) is approximately 0.51/sec (1.95 s / change).

filtering stages are thus employed to mitigate the effects of these local variations.

1) Pre-Filtering: As shown in Fig. 1, pre-filtering is applied directly to chromagrams prior to the pattern matching process. A popular choice is to blur out noisy frames by smoothing the features across neighboring frames, typically using either a moving average filter [11], [18] or a moving median filter [7], [19]. Alternatively, in some systems [1], [20], [21], pre-filtering is done by averaging feature frames between beat positions obtained by beat tracking methods, yielding a so-called beat-synchronous chromagram. In Section V, we present detailed implementations of pre-filtering methods including beat-synchronization.

2) Post-Filtering: In contrast to pre-filtering, post-filtering is performed after the pattern matching stage. With few exceptions [19], [22], [23], post-filtering is mostly applied in the context of hidden Markov model (HMM)-based systems, typically using a Viterbi decoder. An HMM describes the dynamic behavior of features in terms of transitions between finite states (i.e., chords here). The transitions are governed by a set of probabilities called transition probabilities that, in our case, describe the likelihoods of transiting from a chord to another. The Viterbi algorithm finds the most likely chord sequence by evaluating the joint probability of the output from the pattern matching stage and all possible chord sequences based on a given transition matrix. In other words, it can reduce the number of spurious chord predictions by restricting unlikely chord transitions. Consequently, the performance of Viterbi decoding is highly dependent on the transition matrix, and thus many studies have concentrated on finding the optimal setting for these parameters [1], [7], [24], [25]. Among those, the most common approach is to generate the matrix automatically by estimating the transition probabilities from labeled data. Further details about this parameterization strategy and the methodology for evaluating the effects of the resulting matrix are discussed in Section VI.

III. Chroma Features and Variants

In this section, we discuss two of the most commonly applied approaches to improving conventional chroma features. One cleans up chroma features by minimizing the effects of overtone and percussive noise, and the other reduces the feature variance by compressing the dynamic range of the feature. The following subsections discuss the details of these approaches, including the basic calculation of chroma features and the combination of these two different approaches.

A. Basic Chroma Calculation

In this paper, baseline chroma features are calculated using the constant-Q transform [26], one of the most common approaches in the literature. The constant-Q transform is a spectral analysis technique in which a time domain signal $x(n)$ is transformed into its log-frequency domain representation X according to the formula:

$$X(k) = \sum_{n=0}^{N_k-1} w_k(n)x(n)e^{-j2\pi f_k n} \tag{1}$$

where k is the bin position, $w_k(n)$ is a window function of length N_k, and f_k is the center frequency of the kth filter bank. The center frequency f_k follows the frequencies of the equal tempered scale as:

$$f_k = f_{\mathrm{A4}} \cdot 2^{\frac{k-69\beta}{12\beta}} \tag{2}$$

where β is an integer representing the number of bins per semitone, the number 69 is the MIDI note number of A4 (440 Hz), and f_{A4} is the reference tuning frequency of a given song (e.g., $f_{\mathrm{A4}} = 440$ Hz, if a song is perfectly in tune). In practice, f_{A4} is obtained by the tuning method proposed in [27].

We denote the MIDI note number as p, and obtain a 12-bins per octave pitch spectrum $P(p)$ by summing[3] adjacent bins of $|X(k)|$ using a Gaussian window centered at $k = \beta \cdot p$ as:

$$P(p) = \sum_{k=\beta\cdot p-\delta}^{\beta\cdot p+\delta} \exp\left(-\frac{(\beta\cdot p-k)^2}{2\cdot(0.2\cdot(\beta-1))^2}\right)\cdot|X(k)| \tag{3}$$

where $\beta > 1$, and $\delta = \lfloor(\beta-1)/2\rfloor$ specifies the adjacent bin range. When $\beta = 1$, $P(p) = |X(p)|$. Finally, the mth chroma vector $C(b,m)$ with indices $b \in [0,11]$ (i.e., semitones from C to B) can be calculated by simply folding the mth pitch spectrum $P(p,m)$ as:

$$C(b,m) = \sum_{p:p\equiv b(\mathrm{mod}\ 12)} P(p,m) \tag{4}$$

In this paper, we use $\beta = 3$, with the analysis performed on the pitch range $p = 21$ (27.5 Hz) to $p = 108$ (4186 Hz), i.e., A0 - C8, which is the pitch range of a piano. However, low frequency bins require large size constant-Q kernels w_k, resulting in a blurring of chord boundaries. Instead, we fix the window length and hop size to 4096 (186 ms) and 2048 (93 ms) samples respectively at a 22050 Hz sample rate, which provide a good trade-off between time and frequency resolution[4]. An alternative solution is introduced in [28], which uses time-frequency reassignment for both high time and frequency resolution. Because this approach is relatively new and thus not common, it was not tested in this paper. An example of the resulting chromagram is shown in Fig. 2(a). In this paper, all feature vectors are normalized by the ℓ^2-norm to make the features robust to dynamics of a signal[5].

B. Overtone Removal

Ideally, a chroma vector would contain only the fundamental frequencies of the notes in an audio signal. However, all musical instruments produce overtones, and thus the chroma features extracted from real-world music are inevitably affected by these overtones. In addition, due to insufficient resolution in low frequencies, low-pitched sounds are hard to clearly detect as a single pitch, but instead tend to present as a blurred pitch across

[3]Summing magnitudes of a $\log_2$-frequency spectrum is a common approach to calculating a chroma vector [1], [18], [19].

[4]In informal experiments, we have found that low temporal resolution has a greater negative effect on chord recognition performance than low spectral resolution. The values 4096 and 2048 were chosen by examining the window lengths (power of two) between 1024 and 16384 with 50% overlap.

[5]Frame-level feature normalization is a common approach in chord recognition studies [9], [13], [21], [29]. In our unofficial experiments, ℓ^2-norm showed the best overall performance among ℓ^1-, ℓ^2-, and ℓ^∞-norms.

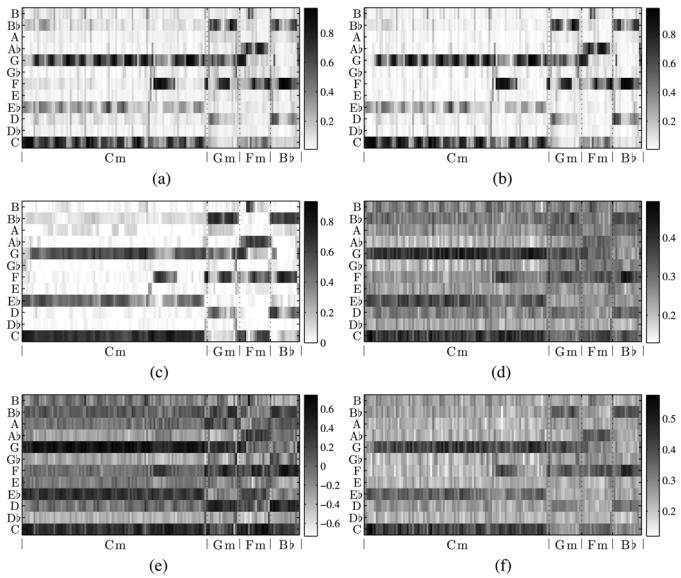

Fig. 2. Various types of chromagrams extracted from the first 15 seconds of "Don't Speak" by No Doubt. The x-axis labels the ground truth chord progression and the y-axis represents the pitch classes. (a) C, (b) C^W, (c) C^N, (d) C_{Log}, (e) C_{CRP}, (f) C^W_{Log}.

several semitones in chroma features. The sounds from percussive instruments such as drums also often deteriorate chromagrams, concealing the harmonic structure of pitched instruments. Bass drums are particularly problematic because they often appear in a pitchgram as a particular pitch, due to their relatively simple harmonic content and long duration, in contrast to short, broadband percussive sounds such as the snare drum. Naturally, the distortion caused by overtones, insufficient frequency resolution, and percussive sounds has been regarded as a major problem reducing the discriminating power of chroma features, as documented in many studies [17], [18], [30], [31].

The most common and simple solution is to de-emphasize high and low frequencies from chroma computation. As high-frequency bands are assumed to consist of mostly overtones, and low frequencies may contain the problematic bass drum sounds and noise caused by low frequency resolution, this filtering can reduce the influence of overtones and unwanted noise on resulting chroma features. In this paper, a Gaussian window centered at C4 ($p = 60$) is applied to a pitch spectrum P prior to the chroma folding process as:

$$\hat{P}(p) = \exp\left(-\frac{(p-60)^2}{2\cdot 15^2}\right)\cdot P(p) \tag{5}$$

The Gaussian window emphasizes mid-frequency bands where the fundamental pitches are expected to be located [32] while suppressing overtones and possible noise in high- and low-frequency bands. The chromagram calculated from $\hat{P}$ is denoted as C^W. As shown in Fig. 2(b), the energies of overtones present in Fig. 2(a) (see the B♭ notes in the 'Cm' chord frames) are attenuated in C^W.

In some recent systems [17], [21], percussive sounds are removed from an audio signal using harmonic/percussive sound separation (HPSS), a source separation algorithm proposed by [33]. However, in our experiments, we did not observe any significant effect of HPSS on chord recognition performance, and these results are thus not reported in this paper. This seems to be due to the fact that HPSS has poor performance on separating bass drums from a signal, which is problematic, as mentioned above.

Another variant based on the overtone removal concept is the non-negative least squares (NNLS) chroma proposed by Mauch and Dixon [31]. In their method, the fundamental pitches in a signal are estimated from a log-frequency spectrum. To do this, they manually generate a dictionary of idealized note profiles, each of which consists of pure harmonic partials. Each note profile is placed in the log-frequency domain with geometrically declining harmonic amplitudes. The amplitude of the hth harmonic is modeled as:

$$v_h = s^{h-1} \tag{6}$$

where the parameter $s \in (0,1)$ defines the spectral shape. The fundamental pitches $\mathcal{P}$ in a signal are then estimated by solving a NNLS problem between the note dictionary D and the signal's log-frequency spectrum $\mathcal{X}$ as:

$$\arg\min_{\mathcal{P}} \|\mathcal{X} - D\cdot\mathcal{P}\|^2, \quad \text{subject to : } \mathcal{P}\in\mathbb{R}^d_{\geq 0} \tag{7}$$

where d is the number of note profiles defined in the note dictionary. In practice, in order to reduce the gap between the artificial overtone patterns in the note profiles and the real overtone patterns in the log-frequency spectrum, spectral standardization is applied to the log-frequency spectrum prior to solving (7). Finally, $\mathcal{P}$ is folded into a 12-bin chroma vector.

In this paper, we extract $\mathcal{P}$ using the NNLS Chroma Vamp plugin[6]. We set $s = 0.7$, following the authors' recommendation for popular music. The tuning reference for each song is set to the same value used in (2). In [31], $\mathcal{P}$ is filtered using a Gaussian window in order to remove any remaining noise in high and low frequencies. Since the Gaussian window used in [31] is almost identical to that defined in (5), $\mathcal{P}$ is filtered using (5). The resulting chroma features are denoted by C^N, see Fig. 2(c).

C. Timbre Homogenization

Most real-world music contains complex combinations of sounds produced by multiple instruments in different pitches, volumes, and rhythms. Consequently, chroma features extracted from music vary widely from song to song, generating multiple levels of complexity. The wide variance in features makes it difficult to build an appropriate classifier that covers the whole feature space.

One of the simplest methods for reducing the feature variance is to limit the dynamic range of the features using logarithmic compression. In [17], the authors interpret the effect of logarithmic compression as making the distribution of features more similar to a Gaussian, thus allowing a better approximation with such models. In this study, we use the compression method suggested by [34]. Log-compression is performed on a pitch spectrum prior to the chroma folding process as:

$$P_{\mathrm{Log}}(p) = \log(1+\eta\cdot P(p)/P_{\max}) \tag{8}$$

[6]http://isophonics.net/nnls-chroma

where $P_{\max}$ is the maximum value of P, and $\eta > 1 \in \mathbb{R}$ controls the degree of compression. We fix $\eta = 1000$ because the value showed the overall best performance in our experiments. The chroma features calculated from P_{Log} are denoted as C_{Log} (see Fig. 2(d)).

In order to minimize the influence of timbre change, [35] presents a method to remove the timbral characteristics of chroma features. This method, referred to as chroma DCT-reduced log pitch (CRP), is based on the fact that a large portion of the timbre of a sound is described by the spectral envelope of the sound [36]. Thus, by removing timbre-related information from a pitch spectrum $P(p)$, the resulting chroma features can be more robust to changes in timbre. A timbre invariant chroma vector is calculated through the following steps: First, apply logarithmic compression to $P(p)$ using (8). Second, transform the logarithmized pitch spectrum using the discrete cosine transform (DCT). The lower coefficients of the resulting cepstrum vector are closely related to the spectral envelope [37]. Third, filter out the first ξ cepstrum coefficients. (i.e., the ξ-lowest coefficients are set to zero). Fourth, transform the cepstrum back to the pitch domain using the inverse DCT, and finally project the resulting pitch vector onto a 12-dimensional chroma. In our experiments, we use $\xi = 54$ with a 120-point DCT as suggested by [35]. It is important to note that by removing the DC component from the cepstrum, the resulting chroma vector contains both positive and negative values, as demonstrated in Fig. 2(e).

D. Combinations of Overtone Removal and Timbre Homogenization

The overtone removal and the timbre homogenization techniques described above seem quite contrary to each other, because while the overtone removal techniques focus on de-emphasizing overtones, the timbre homogenization methods boost the intensities of the overtones due to the use of logarithmic compression. However, the combination of these opposing concepts has been used in recent systems [21], [38], and has shown good performance in the 2011 MIREX chord estimation task[7].

The combinations are implemented by simply applying the window function in (5) to each variance-reduced pitch spectrum (i.e., P_{Log} and P_{CRP}) prior to the chroma folding process. The resulting chromagrams are denoted as C^{W}_{Log} and C^{W}_{CRP}, respectively. The chromagram generated by combining P_{Log} and the window function is shown in Fig. 2(f). For NNLS chroma features, the combination is achieved by applying logarithmic compression to the fundamental pitch spectrum $\mathcal{P}$ prior to the chroma calculation. In fact, the standardization used in the NNLS chroma calculation yields a somewhat similar effect to that of log-compression. Thus, the parameter setting used in P_{Log} and P_{CRP} (i.e., $\eta = 1000$) is too large to be suitable for $\mathcal{P}$. We use $\eta = 100$ instead, which we found empirically to give the best results. The resulting chromagram is denoted as C^{N}_{Log}. Notice that the superscript and subscript of each chroma symbol indicate the technique(s) used for overtone removal and timbre homogenization respectively.

[7]http://nema.lis.illinois.edu/nema_out/mirex2011/results/ace/index.html

IV. Chord Models

This section describes the methodology for estimating the parameters of the chord models evaluated in this paper.

A. Hand-Crafted Chord Template

A hand-crafted chord template is created by describing the known note distribution of a chord in a 12-dimensional vector, resulting in the ideal chroma vector that represents a given chord. In a template vector, each component corresponding to a chord-tone[8] is typically set to 1, and the other components are set to 0. For example, the chord template for a C major triad is [1 0 0 0 1 0 0 1 0 0 0 0], where the left to right order of the vector components follows the twelve-tone equal-tempered scale from C. Since the vector contains only binary elements, it is often termed a binary chord template. The fit of a chord template to a given chroma vector is typically obtained by measuring the Euclidean distance between the template and the chroma vector. The chord label of the given chroma vector is then determined by the chord template that minimizes the distance. Because there is no suitable way to build the no-chord template, we assume a chroma frame where the RMS of the original audio frame is under -57 dB to be a no-chord, which we found empirically to give the best results. The binary template is denoted by BT.

B. Gaussian-Based Stochastic Chord Models

More sophisticated chord models are created by defining probability distributions for each chord class. A popular choice is the multivariate Gaussian distribution, which is defined by a mean vector μ, and a covariance matrix Σ. The distribution parameters, μ and Σ, are estimated from labeled training data typically using the Expectation-Maximization (EM) algorithm [39]. Due to the fact that the unimodality of a single Gaussian is insufficient to model the probability distribution of a complex multimodal feature space, Gaussian mixture models (GMMs) are preferred in many recent chord recognition systems [24], [38], [40]. A GMM is simply a multimodal distribution composed of weighted Gaussian components. Different Gaussian components represent more detailed descriptions of each chord in the training data, producing a more accurate fit.

For training, the chromagrams extracted from training data are segmented based on the chord annotations (i.e., 12 major triad, 12 minor triad and no-chord segments). In order to compensate for the limited amount of training data and to give consistency to the same chord quality (i.e., major and minor) regardless of the root positions, all chord segments, except no-chord segments, are transposed to the C-based chords (i.e., C-major and C-minor). These root-normalized chord segments are used to train C-major and C-minor chord models. The trained chord models are then re-transposed to the remaining 11 major and 11 minor chord models to complete a set of 12 major, 12 minor, and no-chord models. For more details about this transposition method, refer to [7], [15]. This training approach reduces the risk of over-fitting (i.e., the situation in which a model becomes

[8]The pitches which make up a chord are called chord-tones, while all other pitches are called non-chord-tones.

too specific to the training data, and thus loses generality) because it increases the number of training samples and reduces the number of parameters to be trained.

In some cases, the EM algorithm may converge to a solution which contains a singular or nearly-singular covariance matrix for one or more Gaussian components. Those components are usually assigned only a small number of data points lying in a lower-dimensional subspace, yielding infinite density. This leads GMM to place excessive weight on few local optima, thus resulting in severe performance degradation. To cope with this problem, we apply covariance regularization after the M-step of the EM algorithm, by adding a small constant value (in this paper 10^{-6}) to the main diagonals of the covariance matrices. This can prevent the Gaussian density going to infinity by setting the lower bounds of the Gaussian width along all dimensions. In addition, this regularization alleviates the risk of over-fitting, leading the models to be less tightly fit to the given observations during the training process.

For the experiments conducted in this study, we prepare various versions of stochastic chord models, each with a different number of Gaussian components with full covariance matrices. The trained chord models using a single Gaussian and GMMs are denoted by G with a suffix indicating the number of Gaussian components (e.g., $G1$ for a single Gaussian and $G25$ for a GMM consisting of 25 Gaussian components). In the experiments, the fit of a chord model to a given chroma frame is measured by calculating the likelihood of the model for the frame. The predicted chord for the given chroma is then the one associated with the chord models with the maximum likelihood.

V. Pre-Filtering Techniques

In this section, we provide details and examples of pre-filtering approaches.

A. Moving Average and Moving Median Filters

Moving average and median filters are calculated by taking the mean or median value of consecutive L frames as:

- Moving average filter

$$\bar{C}(b,m) = \frac{1}{L}\sum_{l=0}^{L-1} C\left(b,\ m+l-\lfloor (L-1)/2 \rfloor\right) \qquad (9)$$

- Moving median filter

$$\tilde{C}(b,m) = \text{median}\left[C(b,\ i),\ C(b,\ i+1),\ldots,C(b,\ j)\right],$$
$$i = m - \left\lfloor \frac{L-1}{2} \right\rfloor, \quad j = m + \left\lceil \frac{L-1}{2} \right\rceil \qquad (10)$$

The effects of both filters are evaluated with respect to different settings of the smoothing filter length L. All stochastic chord models are retrained for each setting of L, and the filtered chromagrams are used for testing. During the experiments, we only use an odd number of frames for the parameter L so that for any given frame, the filter considers the same number of frames before and after. Fig. 3(a) shows a chromagram, while Fig. 3(b) and Fig. 3(c) show its smoothed versions using the moving average and median filters, respectively. Both filters successfully reveal the long term harmonic trend in the original chromagram in Fig. 3(a).

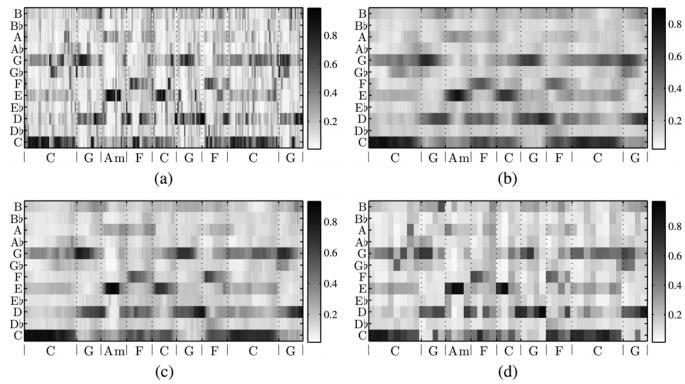

Fig. 3. Chromagrams excerpt from "Let It Be" by The Beatles. (a) Original, (b) Moving average filter ($L = 13$), (c) Moving median filter ($L = 13$), (d) Beat-synchronous chromagram.

B. Beat-Synchronization

Beat-synchronization is performed on a frame-based pitchgram (i.e., a temporal sequence of pitch spectrum vectors) by taking the median (for each pitch bin) of the pitchgram frames between two consecutive beat times. The underlying assumption of this approach is that chords tend to change on beats, and thus the harmonic content between two consecutive beats can be assumed to be uniform. By summarizing feature frames between two consecutive beats, the resulting chroma features can be robust to noise in a music signal by reflecting the long term harmonic trend in the signal. To calculate a beat-synchronous chromagram, we obtain the beat positions of a song using the Tempogram Toolbox[9] [41]. Fig. 3(d) shows a beat-synchronous chromagram which has an effect similar to those of the moving average and median filters. Since beat-synchronization reduces the frame rate, it lowers the computational cost in later processing stages. More importantly, the reduced frame rate may alter the relative importance of other parameters in the subsequent processing stages.

VI. Transition Matrix

As mentioned in Section II-C2, the most widely used approach for building the transition matrix is estimating it from bigrams of symbolic data in the training set. Since the transition matrix should capture the temporal dependence from frame to frame, prior to computing the bigrams, symbolic data is segmented at the same frame rate used in the chromagram calculation. For the same reason, if the chromagrams are beat-synchronized at the pre-filtering step, the symbolic data has to be segmented at the same beat positions used in the beat-synchronization process. Before the segmentation, all chord boundaries in the symbolic data are quantized to the closest beat positions. We denote the frame-based transition matrix and the beat-based transition matrix as A_{F} and A_{B} respectively.

[9]http://www.mpi-inf.mpg.de/resources/MIR/tempogramtoolbox/

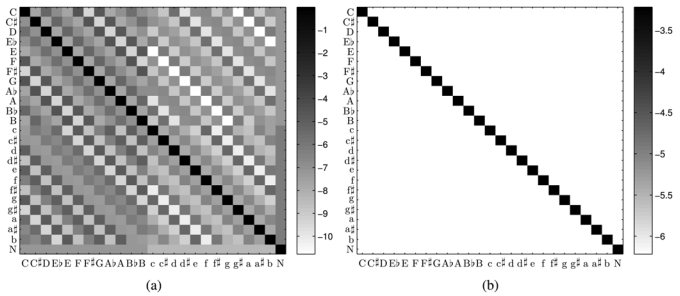

Fig. 4. Examples of transition matrices in a log probability scale: (a) the estimated transition matrix from annotated music data, (b) the uniform transition matrix after applying the transition penalty $\rho = 3$.

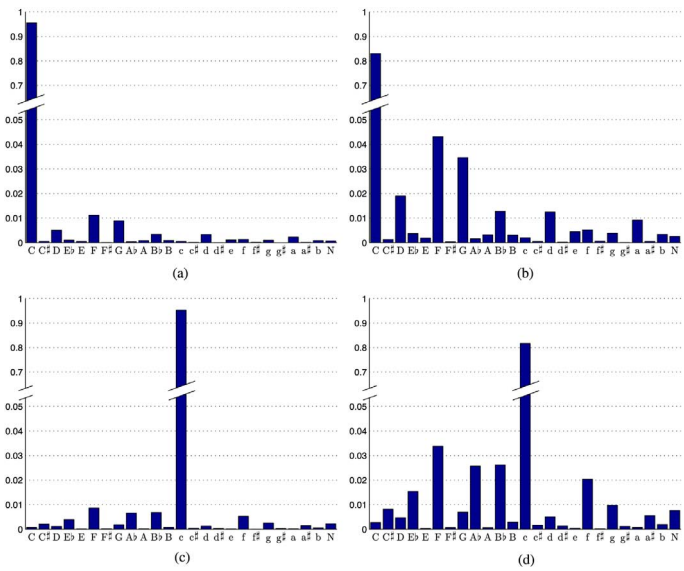

Fig. 5. Transition probabilities from C major triad and C minor triad. (a) A_{F} (C major), (b) A_{B} (C major), (c) A_{F} (C minor), (d) A_{B} (C minor).

As in the case of the chord model training described in Section IV-B, we only consider chord transitions relative to the current chord, i.e., we assume that all transitions happen from a root of C major or C minor. For example, the transitions $\mathrm{C} \rightarrow \mathrm{Am}$ and $\mathrm{E} \rightarrow \mathrm{C}^{\sharp}\,\mathrm{m}$ are both counted as $\mathrm{C} \rightarrow \mathrm{Am}$ (i.e., $\mathbf{I} \rightarrow \mathbf{vi}$)[10]. The root-normalized bigrams are then transposed to the other major and minor roots to form the final matrix. Fig. 4(a) shows an example of the estimated transitions matrix.

Fig. 5(a) and Fig. 5(b) show transition probabilities from the C major triad in A_{F} and A_{B}, respectively. In the figures, the two most common chord transitions[11] in popular music, $\mathrm{C} \rightarrow \mathrm{F}$ and $\mathrm{C} \rightarrow \mathrm{G}$, have relatively higher probabilities than the other transitions except $\mathrm{C} \rightarrow \mathrm{C}$. In contrast, $\mathrm{C} \rightarrow \mathrm{F}^{\sharp}$ is a rare transition[12] in popular music, and thus has a very low probability in both cases.

[10]In this paper, Roman numerals are often used to indicate the harmonic relationship between two chords without reference to actual chord symbols. In this notation, the first seven Roman numerals represent a major scale degree from the root. Capital letters are used for major triads, while lowercase letters are used for minor triads, and a flat ($\flat$) or sharp ($\sharp$) in front of a Roman numeral lowers or raises the diatonic pitch by a half step.

[11]Progressions in which the chord roots ascend by 4th and descend by 5th are the most frequent and strongest in jazz and popular music [42].

[12]This chord progression is commonly avoided due to its augmented fourth root motion that is one of the most dissonant musical intervals [42].

The noteworthy observation in these figures is the extremely high self-transition probabilities, i.e., $\mathrm{C} \rightarrow \mathrm{C}$ in both Fig. 5(a) and (b) and $\mathrm{c} \rightarrow \mathrm{c}$ in both Fig. 5(c) and (d). The self-transition probability represents the probability of staying in the same chord frame to frame. In the case of both frame-based and beat-synchronous analysis, chord durations are typically longer than frame length. Accordingly, the chord remains stable for several frames, making the transition probability to itself the highest. This appears as a strong diagonal in the transition matrix, as seen in Fig. 4(a). While many previous works report improving the accuracy rate with the trained transition matrix, most of them contain little discussion of these self-transition probabilities. They merely explain that the musical context described in the transition matrix reduces chord confusions that occur in the pattern matching stage. Only a few studies including our own [6] and [10] argue that most of the improvement is due to a relatively high self-transition probability, which essentially acts to minimize the number of chord changes.

To verify this argument, we use a transition penalty ρ, which is similar to the insertion penalty used widely in speech recognition [43]. The penalty is a fixed value subtracted from the log probability of each transition but the self-transition, thus reducing the chances of transitions out of the current chord. In brief, this penalty controls the relative strength between the self-transition and the other transitions. It is applied as follows:

$$\log(\hat{a}_{i,j}) = \begin{cases} \log(a_{i,j}) - \rho & \text{for} \quad i \neq j \\ \log(a_{i,j}) & \text{for} \quad i = j \end{cases} \tag{11}$$

where $A = [a_{i,j}]$ is the original transition probability matrix and $\hat{A} = [\hat{a}_{i,j}]$ is the modified matrix with penalty ρ. In addition, to evaluate the effect of the transition probabilities other than the self-transition probability, we define a uniform transition matrix A_{U} in which all transitions have the same probability. With a high penalty value, A_{U} minimizes the number of chord transitions without considering musical context. Fig. 4(b) shows an example of A_{U} after applying the transition penalty $\rho = 3$. In order to apply post-filtering to the output of BT, pseudo-probabilities are calculated by taking the reciprocal of the Euclidean distances between chromagram frames and the chord templates.

VII. Experiments

The experiments are conducted on all possible combinations of the stages as follows:

Expt1: *Feature extraction and pattern matching*
In this experiment, we assess the performance of different combinations of the chroma features and chord models described in Section III and Section IV.

Expt2: *Effect of pre-filtering*
In this experiment, we evaluate the effects of pre-filtering approaches described in Section V.

Expt3: *Effect of post-filtering*
In this experiment, we probe into how the musical contexts represented in a transition matrix impact chord recognition performance using the transition penalty ρ.

Expt4: *Combined pre- and post-filtering*
In this experiment, we investigate the relationship between pre- and post-filtering and evaluate all possible parameter combinations.

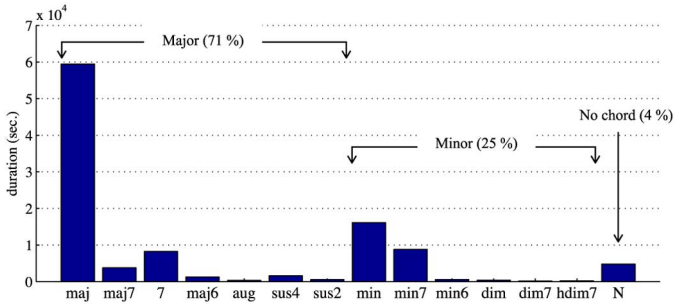

Fig. 6. The distribution of chord types in the data set.

All experiments above are performed on 495 chord annotated songs. The data set comprises 180 songs from Christopher Harte's Beatles dataset[13], 20 songs from Matthias Mauch's Queen dataset[13], 100 songs from the RWC (Real World Computing) pop dataset[14] and 195 songs from US-Pop dataset[14]. The RWC and US-Pop datasets were manually annotated by music students at NYU [38]. The distribution of chord types in the data set is presented in Fig. 6. For the major/minor evaluation, we map chord labels that have a minor 3rd as their first interval to minor chords, and all others to major chords, following MIREX evaluations. The resulting distribution consists of 71% major, 25% minor and 4% no chords, as depicted in Fig. 6.

In the case of the stochastic chord models, all the reported results are obtained using 5-fold cross validation, with each group containing 99 songs selected randomly from the data set. For each fold, one group is selected as a test set, and the remaining 4 groups are used for training. The chord recognition rate is calculated as follows:

$$\text{Accuracy} = \frac{\text{total duration of correct chords}}{\text{total duration of dataset}} \times 100\% \quad (12)$$

VIII. Results and Discussions

A. Expt 1: Feature Extraction and Pattern Matching

The results are shown in Table I. In the table, the baseline results of C are placed in the top row, followed by the results of the chroma features based on overtone removal (C^W and C^N), timbre homogenization (C_{Log} and C_{CRP}), and their combination (C^W_{Log}, C^W_{CRP} and C^N_{Log}). The last column of the table is the difference between the maximum and baseline (BT) values of each row, indicating how much impact model complexity has on chord recognition performance. Similarly, the last row shows the impact of the different chroma features. Overall, while increasing the model complexity does not show a significant impact on performance (average 1.59%), the changes in chroma features have a great impact on performance (average 9.79%).

1) Effect of Chord Model Complexity: It would be reasonable to expect that performance would improve with increasing complexity of the chord model. However, the results in Table I contradict this expectation. In the table, the best result is obtained using $G1$, which is the simplest stochastic chord model. Moreover, in the case of C^W, all stochastic chord models perform worse than the simplest, BT. In other words, increasing model complexity does not guarantee better performance in this experiment. One possible reason for this unexpected phenomenon is over-fitting. However, when we performed the same experiments on training data, the results produced the same behaviors shown in Table I. In addition, although the major chord model was trained using almost three times as much data as the minor chord model, in our experiments, an increase in model complexity degraded performance for the major chord model but improved performance for the minor chord model (this phenomenon will be addressed in the latter part of this section). Therefore, over-fitting is not a plausible explanation. Instead, the reason for this phenomenon can be found by examining the distribution of chord classification errors for each chord model.

TABLE I
Average Accuracy Without Filtering

	BT	$G1$	$G5$	$G10$	$G15$	$G20$	$G25$	max - BT
C	46.95	46.46	46.26	48.10	48.39	48.74	48.77	1.82
C^W	52.12	49.40	47.38	50.24	51.04	51.42	51.71	0
C^N	54.38	54.51	50.49	50.90	51.42	52.14	51.97	0.13
C_{Log}	45.18	48.24	50.44	49.34	49.36	49.06	49.35	5.26
C_{CRP}	44.37	40.12	39.80	39.61	40.49	40.87	40.86	0
C^W_{Log}	55.51	**58.30**	57.58	57.73	57.72	57.70	57.69	2.79
C^W_{CRP}	53.24	53.83	54.66	54.67	54.00	54.03	53.55	1.43
C^N_{Log}	55.00	56.29	53.09	53.24	53.28	53.33	53.43	1.29
max - C	8.56	11.83	11.32	9.63	9.33	8.96	8.92	

The distribution of chord classification errors for C^W is shown in Fig. 7. C^W is selected because it is the most egregious case. In this figure, the results are key-normalized to major (top) and minor (bottom) triads and averaged across all roots. In Fig. 7(a), the label **I** represents correct classification and the remaining labels represent mis-classified chords in Roman numeral notation. Similarly, in Fig. 7(b), the label **i** represents correct classification and other labels represent mis-classified chords. As depicted in these figures, the majority of confusions occur between harmonically related chords, such as chords in a parallel relationship (i.e., **I** and **i** in both Fig. 7(a) and 7(b)) and the chords in a relative relationship (i.e., **I** and **vi** in Fig. 7(a), ♭**III** and **i** in Fig. 7(b)). These chord pairs share at least one note in common with each other. For example, **I** and **i** share a common root and fifth, and have only a semi-tone difference between the thirds. Due to the sharing of notes, the chroma vectors from these harmonically related chords are close to each other. Thus, the chord models estimated from those "close" data samples become significantly overlapped with each other, inducing confusions in classification. Consequently, the chord classification performance is highly dependent on how the decision boundaries are formed in these overlapping areas between competing chords.

Unfortunately, for GMM-based classifiers trained using EM, the decision boundaries are not optimized by considering the distribution of the entire data space. Instead, they are formed from individually trained GMMs. For example, in our case, each GMM is trained independently for a particular chord with only the data belonging to that chord, as described in Section IV-B. Since the relative probability densities between competing

[13]available from http://www.isophonics.net/datasets

[14]available from https://github.com/tmc323/Chord-Annotations.git

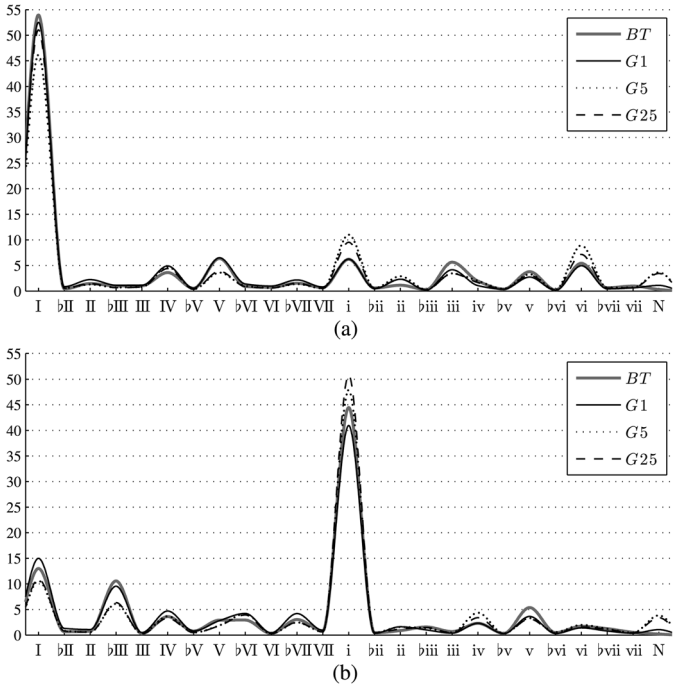

Fig. 7. Chord confusion distributions in the case of C^W from chord models of various complexities: the labels **I** in (a) and **i** in (b) represent correct classifications and the remaining labels represent mis-classified chords. (a) Major, (b) Minor.

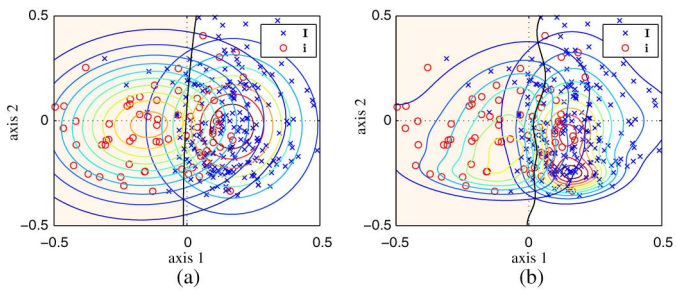

Fig. 8. The distributions of the major (**I**) and minor (**i**) chord models of various complexity. (a) $G1$, (b) $G5$.

chords are not considered when their decision boundaries are formed, these boundaries can be suboptimal. Furthermore, although increasing the model complexity will help each GMM describe the distribution of a particular chord well, it will not guarantee optimal classification performance because the resulting GMMs are still independent from each other. In fact, increasing the model complexity can worsen the performance, as seen in Table I.

An example of the phenomenon that a higher model complexity does not ensure a better result is shown in Fig. 8. This example illustrates how the decision boundary between two chord models, **I** and **i** trained using C^W, changes with different model complexities. For visualization purposes, the example chroma samples and the distributions of the chord models are projected on the axes found using linear discriminant analysis[15] (LDA). Comparing Fig. 8(a) (the case of $G1$) and Fig. 8(b) (the case of $G5$) shows that, since each chord model is unaware of the competing chord model, as described above, the decision boundary

[15]LDA finds the projection that maximizes inter-class separability while minimizing intra-class variability.

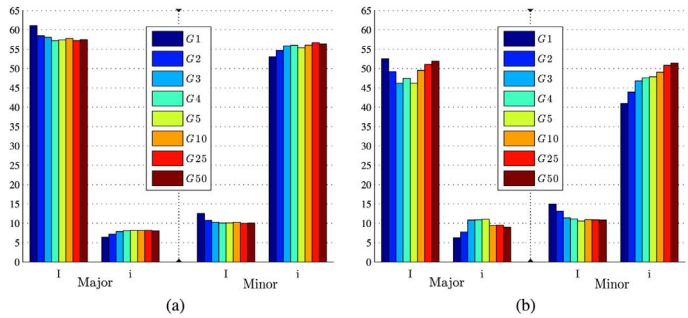

Fig. 9. Chord classification results from chord models of various complexities: (a) confusions between **I** and **i** for major chord (left half) and minor chord (right half) detection in the case of C^W_{Log}, and (b) in the case of C^W.

can move away from the minority class region (i.e., **i**) and spread into the majority class region (i.e., **I**). As a result, $G5$ yields lower classification accuracy on **I** and higher classification accuracy on **i**, compared to $G1$. This is visible in Fig. 7. Due to the imbalanced distribution of chord types (i.e., 71% major and 25% minor) as seen in Fig. 6, the lowered accuracy of the major model is more crucial than the improved accuracy of the minor model to the overall performance. This explains why the performance of $G5$ is lower than that of the simpler $G1$, as shown in the second row of Table I. Similarly, in Fig. 7, even if $G25$ largely outperforms BT on minor chord detection, this is not reflected in the total overall results shown in Table I, but rather the relatively small decrease in performance from BT to $G25$ on major chord detection contributes more to the average performance.

2) Effect of Various Types of Chroma Features: Compared to the results of C, overtone removal (i.e., C^W and C^N) significantly increases chord recognition performance for all chord models. However, the effect of timbre homogenization is marginal (C_{Log}) or even negative (C_{CRP}). In fact, the logarithmic compression used in the timbre homogenization methods not only suppresses the spiky pitches in the pitch spectrum but also boosts the intensities of overtones and any noise present in the signal. As described in Section III-B, these overtones and noise deteriorate the quality of chroma features. In the case of C_{Log} and C_{CRP}, the effect of overemphasized overtones overwhelms the effect of timbre homogenization, leading to lowered performance. This limitation can be overcome by combining the timbre homogenization techniques with the overtone removal techniques. As shown in rows 6 - 8 of Table I, all features based on the combination of overtone removal and timbre homogenization largely increase the accuracy for all chord models.

In the case of the stochastic chord models, an important benefit of using the timbre homogenization technique is that it increases the computational efficiency of the chord modeling process. Due to the reduced feature variability, each chord model requires a relatively small number of parameters (i.e., a small number of Gaussian components) to adequately describe chord distributions. Fig. 9 shows the confusions between two chords in a parallel relationship (i.e., **I** and **i**) for chord models with various complexities. For major chord detection (presented in the left half of each graph), the label **I** indicates correct classifications and **i** indicates that "I" is misclassified as "i". For minor chord detection (the right half of each graph),

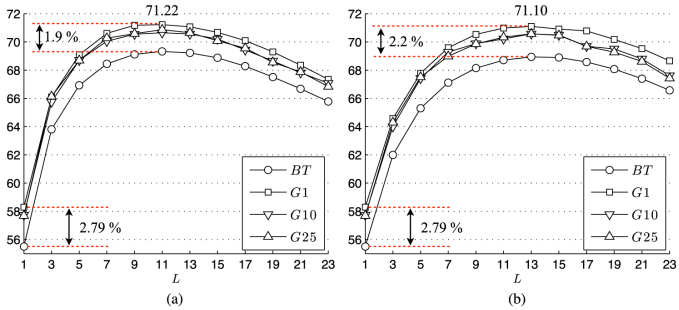

Fig. 10. Average chord detection accuracy as a function of pre-filter order L in the case of C^W_{Log}. $L = 1$ means "no filtering". (a) Moving Average Filter, (b) Moving Median Filter.

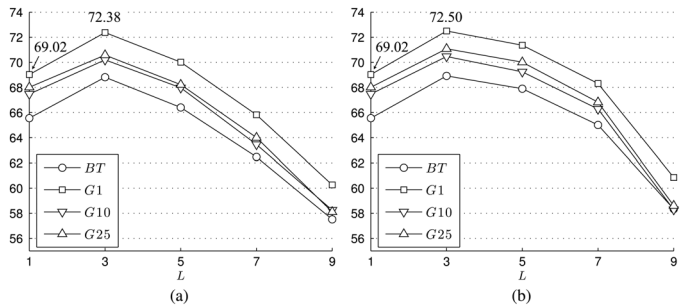

Fig. 11. Average chord detection accuracy as a function of pre-filter order L in the case of beat-synchronous C^W_{Log}. The value at $L = 1$ shows the result from beat-synchronization without any additional filtering. (a) Moving Average Filter, (b) Moving Median Filter.

the label **i** indicates correct classifications and **I** indicates mis-classifications ("i" as "I"). As Fig. 8 shows, when using timbre homogenized features C^W_{Log}, models with complexities greater than $G2$ do not appreciably change either the recognition rate or the confusion rate. This implies that a small number of Gaussians (two in this example) are already sufficient to describe the distribution of a chord, and thus adding more Gaussian components does not significantly alter the probability distribution of each chord model. On the contrary, in the case of C^W, the probability distribution of each chord model can be altered greatly by adding more Gaussian components due to its higher feature variability, see Fig. 9(b).

An interesting aspect shown in Table I is that although the overtone removal mechanism of C^N_{Log} is more complex than that of C^W_{Log}, the performance of C^N_{Log} does not outperform that of C^W_{Log}. Particularly for the stochastic chord models, the performance differences between these two features are substantial, while there is no significant difference with the binary template BT. As described in Section III-B, the NNLS method uses artificial tone profiles to estimate the fundamental pitches in a given audio signal. These tone profiles consist of pure harmonic partials with predefined amplitudes. Due to the fact that not all musical instruments have overtone patterns that exactly match the artificial tone profiles, the NNLS method sometimes fails to detect the fundamental pitches of a given signal or may underestimate their energies. In the case of BT, the effect of these erroneous chroma vectors is limited to isolated frames of the chromagram. However, in the case of stochastic chord models, these inaccurate chroma vectors can severely degrade the quality of chord models when used as training samples. The degraded models adversely affect overall performance, as demonstrated in the row of C^N_{Log} in Table I.

In Table I, the additional cepstral filtering process used in C_{CRP} and C^W_{CRP} calculations does not positively affect the chord recognition performance, as compared to the original C_{Log} and C^W_{Log}; rather, cepstral filtering significantly worsens the performance. This trend is almost perfectly maintained across all experiments, therefore we do not report results of either C_{CRP} or C^W_{CRP} in the remaining experiments.

B. Expt 2: Effect of Pre-Filtering

In this experiment, all pre-filtering techniques (i.e., using moving average / median filters, and the beat-synchronization) largely improve performance over the results in Table I by approximately $13\sim 15\%$. Since the effect of pre-filtering is quite consistent for all chroma feature and chord model pairs, we do not report their detailed results here.[16] Instead, we discuss the effects of pre-filtering by examining the case of C^W_{Log}, which shows the best performance in all experiments.

1) Moving Average and Median Filters: The effects of different filter orders L on the performance using C^W_{Log} are shown in Fig. 10. As seen in the figure, there is no significant difference between the performance of the moving average and median filters. The accuracy versus L curves of the moving median filter are a little slower than corresponding curves of the moving average filter. In other words, the moving median filter needs a slightly higher filter order to achieve its optimal result, and its value is almost identical to that of the moving average filter. For both filters, the relative performance of the different chord models is almost identical across values of L. For example, as indicated in Fig. 10, the initial performance gap between BT and $G1$ without filtering (i.e., 2.79% at $L = 1$) remains unchanged by different values of L; compared to the gaps at the optimal L values of moving average (1.9% at $L = 11$) and median (2.2% at $L = 13$) filters, the changes are both less than 1%.

2) Beat-Synchronous Chromagram: Beat-synchronization is expected to have an effect similar to that of the moving average or moving median filters, because it removes transient noise in a fashion similar to the aforementioned filtering methods. As expected, the recognition rates in this case are approximately 10% higher than the case of unfiltered chromagrams seen in Table I. The highest value is 69.02% with the pair of C^W_{Log} and $G1$ (see Fig. 11 at $L = 1$). However, the effects of beat-synchronization are inferior to the effects of pre-filtering using the moving average or moving median filters, which show maximum 71.22% and 71.10% accuracy respectively.

In fact, most existing beat-tracking algorithms, including the algorithm used in this experiment, tend to overestimate the actual beat-rate by a factor of 2 or 3. Because of the resulting over-segmentation, the number of frames in each beat segment is relatively small and inadequate to cope with the transient noise in a signal. Thus, an additional smoothing process using a moving average or median filter can increase the performance. The results are shown in Fig. 11. As shown in this figure, the

[16]The detailed results are available on https://files.nyu.edu/tmc323/public/taslp/results.html

TABLE II
AVERAGE ACCURACY WITH POST-FILTERING

	(a)					(b)				
	BT	$G1$	$G5$	$G10$	$G25$	BT	$G1$	$G5$	$G10$	$G25$
C	66.0 (2.1)*	63.7 (14)	69.1 (17)	71.0 (17)	71.8 (16)	66.4 (-3.1)	64.3 (8)	69.6 (11)	71.5 (11)	72.2 (10)
C^W	70.6 (2.4)	66.6 (15)	71.1 (20)	73.1 (19)	74.4 (16)	70.9 (-3.0)	67.1 (8)	71.5 (13)	73.6 (12)	74.8 (10)
C^N	68.7 (4.7)	68.6 (18)	70.4 (21)	71.9 (19)	72.8 (20)	69.1 (-0.9)	68.9 (13)	70.7 (16)	72.7 (14)	73.1 (14)
C_{Log}	59.6 (0.2)	61.9 (7)	69.0 (10)	66.4 (8)	65.9 (7)	55.7 (-4.4)	62.6 (1)	69.5 (3)	67.1 (1)	66.6 (0)
C^W_{Log}	73.0 (0.4)	75.4 (10)	77.3 (13)	77.4 (12)	**77.5** (11)	66.2 (-4.3)	75.8 (4)	77.6 (7)	77.8 (6)	**77.9** (4)
C^N_{Log}	70.1 (4.1)	72.4 (16)	72.6 (18)	73.1 (18)	73.5 (18)	70.8 (-2.1)	72.7 (11)	72.9 (11)	73.4 (11)	73.8 (12)

*The optimal penalty value ρ for each result is given in parentheses.

additional smoothing process improves performance to levels comparable or superior to the results in Fig. 10. In this experiment, the optimal L values are all the same at $L = 3$ regardless of the different feature types and model complexities. Considering the shapes of the accuracy versus L curves in both Fig. 10 and Fig. 11, the optimal L value clearly depends on the frame rate, rather than on the feature type or model complexity.

C. Expt 3: Effect of Post-Filtering

The results of post-filtering and the corresponding optimal ρ values are summarized in Table II. Overall, post-filtering shows a significant improvement over the results in Table I, especially in the case of the stochastic chord models. Surprisingly, when compared with the result of A_{U}, the musical context information from A_{F} only improves the accuracy marginally. The largest improvement is only about 0.8%, and in the cases of BT with C_{Log} and C^W_{Log}, A_{F} adversely influences the accuracy.

Unlike pre-filtering, for a given feature set except C_{Log}, performance of a stochastic chord model consistently improves with increasing the number of Gaussian components, and the maximum accuracy is achieved by the most complex chord model. In the case of pre-filtering, the classification performance is entirely dependent on decision boundaries formed between chord models. As we already mentioned, a higher model complexity does not ensure better decision boundaries, and thus the results of pre-filtering did not show any explicit trends with changes in model complexity. However, post-filtering is performed based on the probability values from different models. Therefore, the performance of post-filtering is highly related to the quality of probability estimates rather than the decision boundaries of the models. In most cases, chord models with higher complexity better represent their corresponding chord distributions, resulting in improved performance with post-filtering, as shown in Table II. In the exceptional case of C_{Log}, additional Gaussians are used to describe its overemphasized overtones and noise. Since the overtones produce confusion with other chords, and noise is shared by all chord samples, the results from complex models can be worse than for simpler chord models.

The optimal penalty values vary widely depending on different types of features and model complexities. Fig. 12 shows the effect of the penalty on different combinations of chord models, features, and transition matrices. These combinations are labeled S1 through S6 for convenience. In this figure, the plots of BT (see S1 and S2) have very steep curves and peak at smaller settings than other stochastic chord models. In other

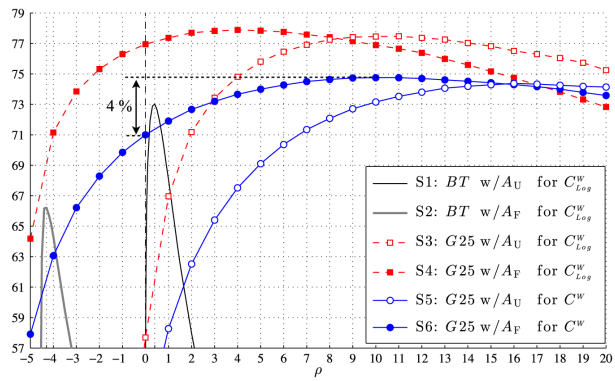

Fig. 12. Average accuracy as a function of ρ.

words, these systems are overly sensitive to the transition penalty. When using BT, the likelihoods of each chord class tend to be very close together, such that the likelihoods are easily overwhelmed by the transition probability depending on ρ. In the case of using A_{F}, the high self-transition probabilities presented in the transition matrix already exceed the discriminating power of BT. Therefore, the effect of these self-transition probabilities needs to be lowered to maximize performance. This results in the peak at $\rho = -4.3$ in Fig. 12 (see S2). Note that the optimal ρ values in the BT column of Table II(b) are all negative. In addition, in the case of C_{Log} and C^W_{Log} in the BT column, the large negative penalties overemphasize the off-diagonal transition probabilities in A_{F}, leading to deteriorated performance compared to using A_{U}.

Interestingly, although many systems use the transition matrix obtained from labeled data (i.e., A_{F}) without any changes, Fig. 12 shows the performance of the transition matrix can be improved by simply adjusting the self-transition probabilities. For instance, when the optimal penalty value ($\rho = 10$) is applied to the transition matrix A_{F}, the accuracy of S6 is improved by almost 4%, as indicated in Fig. 12 (i.e., when the self-probabilities become higher than the original at $\rho = 0$). In fact, since A_{F} is estimated using only the counts of chords in the training set, it is fixed regardless of the choice of feature type and model parameters. Meanwhile, the likelihood values from chord models vary widely, depending on the choice of feature type and model parameters. As there is no guarantee that A_{F} and chord models yield scores on the same scale, the trained matrix can produce suboptimal performance, and thus requires adjusting parameters such as ρ. This is a well known issue in the speech recognition community [43]–[45].

Another noteworthy observation in Fig. 12 is that, for a given chord model and feature set, not only the maximum accuracy but also the shapes of the accuracy versus penalty curves are nearly identical regardless of the choice of transition matrix (A_{U} or A_{F}). For example, the two dotted curves for S3 and S4, which use the same chord model $G25$ and feature C^{W}_{Log}, but different transition matrices, match up quite well if the curve for S3 is moved about 6 units to the left. Similarly, in the case of S5 and S6, the effect of A_{F} is merely a shift of the curve for S5 to the left by about 6 units. This trend is also evident in Table II where the gaps between the optimal penalties for A_{U} and the corresponding penalties for A_{F} are fairly constant at around 6. This number is directly related to the high self-transition probability values given in A_{F}. As seen in Fig. 5(a) and Fig. 5(c), the self-transition values for major triad, minor triad, and no-chord are approximately 95.5%, 95.2% and 98.8%, respectively. When $\rho = 6$ is applied to A_{U}, the self-transition probability for all chord types becomes 94.4%. In other words, A_{U} and A_{F} have a similar effect on chord recognition performance when their self-transition probabilities are roughly equal to each other by adjusting the transition penalty ρ. Therefore, it can be concluded that most of the effects of post-filtering are highly dependent on self-transition probability values, and thus, the influence of other-transition probabilities are almost negligible. This fact is consistent with the results described in [6], [10].

The optimal ρ value is closely related to the number of chord changes in the test set. Fig. 13 shows the ratio of the total number of chord changes in the estimated chord sequence to that in the ground truth sequence as a function of model complexity (for each model, this ratio is computed from sequences estimated using the optimal ρ value). This relationship is shown for different transition matrices and features. As shown in the figure, the best results are in general achieved when the number of estimated chord changes is roughly equal to that of the ground truth chord changes, i.e. when the ratio is close to 1. In the case of the simpler chord models, however, the best results are obtained when the number of estimated chord changes is greater than that of the ground truth changes, i.e. when the ratio is greater than 1. This is likely caused by the fact that simple, and thus unreliable, chord models can yield relatively long and incorrect chord estimations with a low chord change rate. Thus, this phenomenon is more apparent in the case of chord models using C^{W}, which have relatively poor performance, than in case of the models using C^{W}_{Log} (see $G1$ to $G5$ in Fig. 13).

D. Expt 4: Combined Pre- and Post-Filtering

1) Moving Average Filter with Post-Filtering: The results of a moving average filter and post-filtering combination are presented in Table III. The case of using a moving median filter is not reported because its outcomes are nearly identical to other results in the table. Similarly, we only report the results of using A_{F}, because there are very small differences between A_{F} and A_{U}, as described in the previous section. Only for two exceptional cases, the BT/C_{Log} and BT/C^{W}_{Log} pairs, the results from using A_{U} are reported. This is due to the fact that A_{F} has a significant negative effect on these pairs for the reason explained in the previous section (i.e., the over-emphasized off-diagonal probabilities).

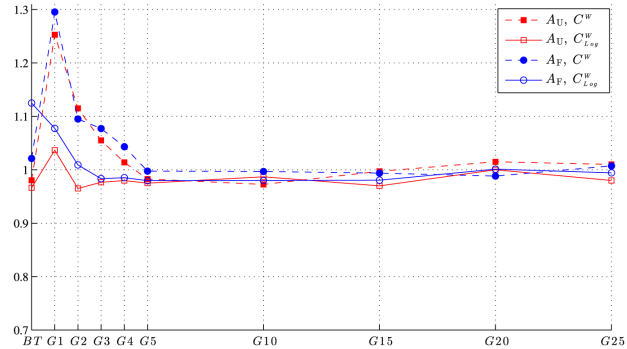

Fig. 13. The ratio of the total number of chord changes in the estimated chord sequence to that in the ground truth sequence, as a function of model complexity for different features and transition matrices.

TABLE III
AVERAGE ACCURACY WITH COMBINED PRE- AND POST-FILTERING USING A MOVING AVERAGE FILTER AND A_{F}

	BT	$G1$	$G5$	$G10$	$G25$
C	66.71 * (3, -3.4)	66.07 (5, 8)	69.34 (3, 14)	70.94 (3, 13)	71.95 (3, 12)
C^{W}	71.08 (3, -3.2)	68.71 (5, 8)	71.61 (3, 14)	73.69 (3, 14)	74.58 (3, 13)
C^{N}	69.21 (3, -1.3)	69.54 (5, 15)	71.05 (3, 15)	71.78 (3, 14)	72.51 (3, 14)
C_{Log}	59.68 † (3, 0.2)	64.05 (3, 1)	69.14 (3, 2)	65.69 (3, 0)	69.94 (3, 1)
C^{W}_{Log}	73.13 † (3, 0.3)	75.58 (3, 6)	76.58 (3, 7)	77.35 (3, 6)	77.58 (3, 5)
C^{N}_{Log}	70.81 (3, -2.2)	72.74 (3, 13)	73.45 (3, 12)	73.48 (3, 10)	73.63 (3, 12)

* The optimal parameters are presented in parentheses as $(L,\ \rho)$
† These results are obtained by using A_{U}

The majority of the best results occur using the minimum pre-filtering parameter (i.e., $L = 3$). Compared to the results of post-filtering alone in Table II, the additional pre-filtering slightly increases the accuracy of simpler chord models, e.g., the systems using BT and $G1$. However, for the more complex chord models, the effects are insignificant or even negative. Particularly, for the best performing feature set C^{W}_{Log}, the combination of both filtering stages does not have any positive effect on performance. The moving average and median filters not only smooth out noisy frames but also blur chord boundaries and informative details in a signal. This may not cause a problem for BT and $G1$ because the quality of these chord models are invariant or insensitive to the details of sample distribution. However, the smoothed samples can deteriorate the quality of complex chord models. As the performance of post-filtering is highly dependent on the quality of the chord models, the additional pre-filtering can hurt overall performance compared to that of post-filtering alone. For more details about this problem, we refer the reader to [38].

2) Beat-Synchronization with Post-Filtering: Table IV shows the results of post-filtering applied to beat-synchronous chromagrams. In this experiment, applying either a moving average or median filters to the beat-synchronous chromagram decreased the performance. The results follow a similar pattern as those shown in Table II, which are the results of post-filtering applied to frame-based chromagrams. The accuracy of all feature/model pairs, with the exception of BT, are increased when A_{B} is used for decoding instead of A_{U}. The performance gaps between A_{U}

TABLE IV
AVERAGE ACCURACY WITH A COMBINATION OF BEAT-SYNCHRONIZATION AND POST-FILTERING.

	(a)					(b)				
	BT	$G1$	$G5$	$G10$	$G25$	BT	$G1$	$G5$	$G10$	$G25$
C	54.4 (0.5)*	60.5 (3)	64.3 (5)	65.5 (5)	66.3 (5)	52.6 (-2.8)	61.5 (-1)	65.6 (0)	66.6 (0)	67.5 (0)
C^W	67.9 (0.6)	67.1 (4)	69.5 (6)	71.7 (6)	72.6 (6)	64.3 (-2.7)	67.9 (0)	70.4 (1)	72.5 (1)	73.4 (1)
C^N	69.2 (0.7)	66.7 (5)	68.6 (6)	70.8 (6)	71.6 (6)	65.7 (-2.7)	67.7 (0)	69.9 (2)	72.0 (2)	72.7 (1)
C_{Log}	53.7 (0.1)	59.1 (2)	62.8 (2)	66.4 (2)	68.2 (2)	44.4 (-3.0)	60.4 (-2)	64.5 (-2)	68.0 (-2)	69.4 (-2)
C^W_{Log}	72.7 (0.1)	76.2 (3)	76.5 (4)	76.1 (3)	**76.7** (3)	55.9 (-3.0)	76.7 (-1)	77.3 (-1)	77.0 (-1)	**77.5** (-1)
C^N_{Log}	70.5 (0.2)	73.2 (3)	74.2 (5)	74.8 (5)	74.9 (4)	59.2 (-2.9)	74.0 (-1)	75.2 (0)	75.7 (0)	75.9 (0)

* The optimal penalty value ρ for each result is given in parentheses.

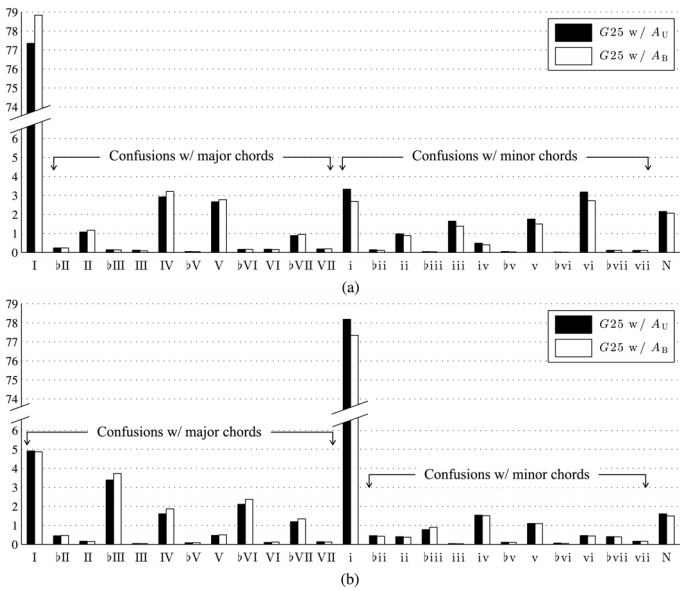

Fig. 14. Comparisons of the classification errors of $G25$ for C^W_{Log} between the case of using A_{U} and the case of using A_{B}. (a) Major, (b) Minor.

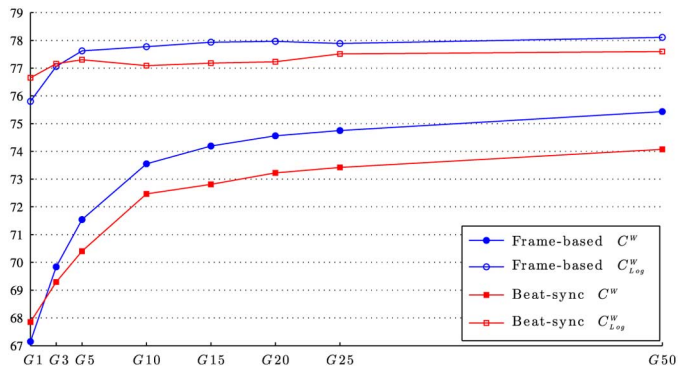

Fig. 15. The best accuracy of chord recognition systems based on different stochastic chord models and different types of chroma features.

and A_{B} are a little larger than the gaps between A_{U} and A_{F} presented in Table II. However, these gaps are still very small (average 1%), showing that the effect of post-filtering is still dominated by self-transition probabilities.

Fig. 14 illustrates the distributions of chord classification errors for systems using A_{U} and A_{B}. When A_{B} is used, the accuracy of major chord detection (i.e., '**I**' in Fig. 14(a)) is improved by approximately 1.5%. However, at the same time, the accuracy of minor chord detection (i.e., '**i**' in Fig. 14(b)) is decreased by about 0.8%. In addition, for both major and minor chord detection, the confusions with major chords are all increased while the majority of confusions with minor chords are decreased. This is due to the fact that, for a trained transition matrix, the probabilities from current chord to major chords are relatively higher than the transition probabilities to minor chords. This can be seen in Fig. 4(a), where the values in the left-half of the matrix (to major chords) are mostly higher than those in the right-half (to minor chords).

Due to the fact that beat-synchronous chromagrams have slower frame-rate than frame-based chromagrams, the optimal penalty values ρ are much smaller than those for frame-based chromagrams. Particularly, for A_{B}, the penalty values are close to zero. In other words, the estimated self-transition probability values are nearly optimal. It seems that because the tempo of all songs is normalized by beat-synchronization, the estimated ratio of frames per chord-change becomes close to the real ratio. In general, the chord durations of slow songs tend to be longer than those of fast songs. In frame-based approaches, a longer chord duration yields a higher self-transition probability. Therefore the estimated self-transition probabilities can be biased to slow songs or fast songs depending on the tempo distribution of songs in a data set. However, this bias can be neutralized by removing the tempo differences between songs using beat-synchronous analysis.

Finally, a noteworthy finding in this experiment, and the experiment in the previous section (i.e., post-filtering on frame-based chromagrams), is that increasing model complexity does not significantly affect the performance of post-filtering when it is applied on the features using the combination of overtone removal and timbre homogenization techniques (i.e., C^W_{Log} and C^N_{Log}). Fig. 15 shows a detailed comparison between the best performances of post-filtering with frame-based and beat-synchronous chromagrams in the case of C^W and C^W_{Log}, varying with model complexity. The features C^W and C^W_{Log} are selected because each shows better results than the other feature in the same feature category (i.e., C^N and C^N_{Log}, respectively). We omit C and C_{Log} due to their low performances. As seen in the figure, for both frame-based and beat-synchronous chromagrams, C^W_{Log} reaches its near-best performance with very few Gaussian components, while C^W needs a large number of Gaussian components until it converges to its best result. This is consistent with the trends shown in Fig. 9. Thus, it can be said that features using the combination of overtone removal and timbre homogenization have benefits in both performance and computational efficiency.

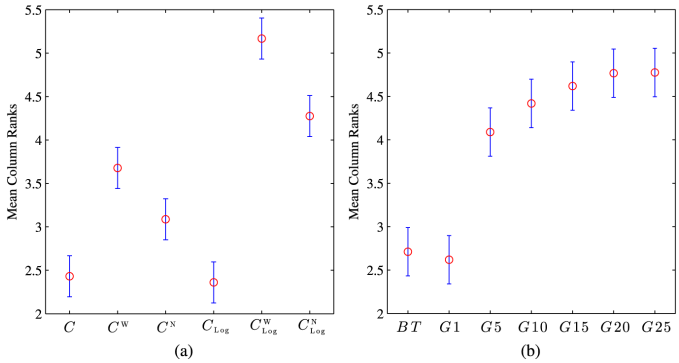

Fig. 16. ANOVA with Tukey-Kramer HSD test ($\mathrm{p} < 0.001$) (a) 15 between the best results for different feature types and (b) 15 between the best results of chord models of different complexity.

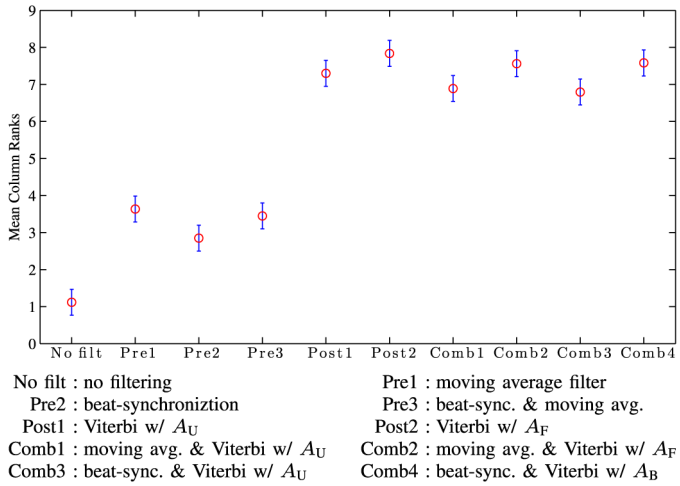

Fig. 17. ANOVA with Tukey-Kramer HSD test ($\mathrm{p} < 0.001$) for different filter combinations in the case of $G25$ for C^{W}_{Log}.

IX. Summary and Conclusion

This paper presents a systematic evaluation of increasingly complex variations of the standard approach to automatic chord recognition. Our experiments demonstrate that the final performance of a chord recognition system is predominantly determined by the front end techniques used in the feature extraction stage. Our results show that the initial performance gap between different features were not fully compensated by any subsequent processing stages. Notably, features using the combination of overtone removal and timbre homogenization techniques show the best results in all experiments. Fig. 16(a) shows the results of a significance test (ANOVA, Tukey-Kramer HSD, $\mathrm{p} < 0.001$) on the differences between the best results for different features obtained in the optimal filter settings, and with $G25$. In this figure, all the differences made by different features are significant except between C and C_{Log}.

Despite the great deal of research investigating how well chord models describe patterns observed in data, we discovered that the benefits of using complex chord models are only valid with post-filtering, and can be largely offset by an appropriate choice of features. Fig. 16(b) shows the significance of differences between chord models having different complexity, in the case of the best performing feature C^{W}_{Log}. As shown in the figure, the effect of model complexity is only significant within few Gaussian components ($G1$ and $G5$) and higher complexities do not yield significant differences. This is consistent with the observation seen in Fig. 15.

In the experiments, we observed that filtering has a very large impact on the accuracy of chord recognition systems. Fig. 17 shows the significance of the differences between various filtering strategies, in the case of using $G25$ and C^{W}_{Log} which produced the best overall performances with filtering. In the case of pre-filtering with either a moving average or median filters, we found that variations of the parameter L have a similar effect across different types of features and different complexities of chord models. Beat-synchronization shows a similar effect to the moving average or moving median filters, but requires an additional smoothing process to compensate for the over-segmentation caused by the beat-detection algorithm. Fig. 17 demonstrates that these pre-filtering techniques (Pre1, Pre2, and Pre3) significantly improve recognition performance compared to the case of no-filtering. Post-filtering has the largest impact, primarily due to the choice of high self-transition probabilities. In Fig. 17, post-filtering (either Post1 or Post2) significantly outperforms pre-filtering. Surprisingly, providing musical context information in the transition matrix (i.e., A_{F} and A_{B}) brings about only marginal improvement over the uniform transition matrix A_{U}, whose self-transition probabilities are roughly equal to those of A_{F} or A_{B}. For instance, in Fig. 17, the effects of using A_{U} (Post1) are not significantly different to those of using A_{F} (Post2). This indicates that any attempts to encode information about likely chord transitions is rendered moot by the need to enforce continuity in the estimations. The combination of pre- and post-filtering leads to relatively small or even negative improvement over the optimal choice of post-filter alone. As shown in Fig. 17, the filter combinations do not show significant difference to post-filtering alone.

As mentioned in Section II-A, most features used for chord recognition do not depart from the conventional form of chroma features. Even a few existing exceptions such as tonal centroid [46] and bass chromagram [31] are also derived from chroma. Since the choice of features appears to be the most important factor in achieving high recognition performance, then the most promising direction for future work is the development of alternative features that are not limited to the concept of chroma. For example in [47], relevant features are learned and obtained from constant-Q spectra using deep learning strategies rather than predefined analytic rules, resulting in state-of-the-art performance.

In this paper, we only investigated the mainstream generative approach, in which chord models and transition matrices are all estimated using the EM algorithm. In this approach, the likelihood of a chord progression is a joint probability distribution, based on the assumption that individual chord events are independent of each other. However, musical events in real-world music including chords are not conditionally independent of each other. Thus there is no guarantee that the generative approach can describe the true distribution of data, and thus may not yield the most optimal outcome. To compensate for this conditional independence assumption, some systems [21], [31]

use multiple transition matrices, forming a complex dependency structure between chords and other music attributes such as key, bass note, and metric position. Discriminative methods, on the other hand, estimate model parameters to maximize the accuracy rate of a given training set directly, without any assumption about the observations. Discriminative approaches were experimentally shown to be superior to generative approaches in a few studies [14], [48], and therefore are worth further investigation.

Finally, it should be noted that the results reported in this paper are optimized to the test set in order to evaluate the effects of various techniques under their optimal parameter settings. Therefore, comparing these results to the results reported elsewhere is unfair. However, the parameters are optimized using cross-validation, thus valid to be used as optimal parameters in future studies.

REFERENCES

[1] J. Bello and J. Pickens, "A robust mid-level representation for harmonic content in music signals," in *Proc. ISMIR*, 2005, pp. 304–311.

[2] K. Lee, "Identifying cover songs from audio using harmonic representation," in *Proc. MIREX Audio Cover Song ID*, 2006.

[3] J. Bello, "Audio-based cover song retrieval using approximate chord sequences: Testing shifts, gaps, swaps and beats," in *Proc. ISMIR*, 2007.

[4] B. Martin, D. G. Brown, P. Hanna, and P. Ferraro, "BLAST for audio sequences alignment: A fast scalable cover identification tool," in *Proc. ISMIR*, 2012.

[5] H. Cheng, Y. Yang, Y. Lin, I. Liao, and H. Chen, "Automatic chord recognition for music classification and retrieval," in *Proc. ICME*, 2008, 2006, pp. 1505–1508.

[6] T. Cho, R. Weiss, and J. Bello, "Exploring common variations in state of the art chord recognition systems," in *Proc. SMC Conf.*, Barcelona, Spain, Jul. 2010, pp. 1–8.

[7] H. Papadopoulos and G. Peeters, "Large-scale study of chord estimation algorithms based on chroma representation and HMM," in *Proc. Int. Workshop CBMI*, 2007, pp. 53–60.

[8] M. Stein, B. M. Schubert, M. Gruhne, G. Gatzsche, and M. Mehnert, "Evaluation and comparison of audio chroma feature extraction methods," in *Proc. 126th Conv. AES*, May 2009.

[9] N. Jiang, P. Grosche, V. Konz, and M. Müller, "Analyzing chroma feature types for automated chord recognition," in *Proc. 42nd Conf. Semantic Audio*, Jul. 2011, AES.

[10] J. Pauwels and J.-P. Martens, "Integrating musicological knowledge into a probabilistic framework for chord and key extraction," in *Proc. Audio Eng. Soc. Conv. 128*, 2010, AES.

[11] T. Fujishima, "Realtime chord recognition of musical sound: A system using common Lisp music," in *Proc. ICMC*, Beijing, China, 1999.

[12] G. Wakefield, "Mathematical representation of joint time-chroma distributions," in *Proc. SPIE Int. Symp. Opt. Sci., Eng., Instrum.*, 1999, vol. 99, pp. 18–23.

[13] J. A. Burgoyne, L. Pugin, C. Kereliuk, and I. Fujinaga, "A cross-validated study of modelling strategies for automatic chord recognition in audio," in *Proc. ISMIR*, 2007, pp. 251–254.

[14] A. Weller, D. Ellis, and T. Jebara, "Structured prediction models for chord transcription of music audio," in *Proc. ICMLA*, Miami, FL, USA, 2009, pp. 590–595.

[15] A. Sheh and D. Ellis, "Chord segmentation and recognition using EM-trained hidden Markov models," in *Proc. ISMIR*, 2003, pp. 185–191.

[16] K. Lee and M. Slaney, "Automatic chord recognition from audio using a supervised HMM trained with audio-from-symbolic data," in *Proc. 1st ACM Workshop AMCMM*, 2006, pp. 11–20.

[17] Y. Ueda, Y. Uchiyama, T. Nishimoto, N. Ono, and S. Sagayama, "HMM-based approach for automatic chord detection using refined acoustic features," in *Proc. ICASSP*, 2010, pp. 5518–5521.

[18] K. Lee, "Automatic chord recognition from audio using enhanced pitch class profile," in *Proc. ICMC*, New Orleans, USA, 2006.

[19] C. Harte and M. Sandler, "Automatic chord identification using a quantised chromagram," in *Proc. 118th Conf. AES*, Barcelona, Spain, 2005.

[20] J. Reinhard, S. Stober, and A. Nurnberger, "Enhancing chord classification through neighbourhood histograms," in *Proc. Int. Workshop CBMI*, 2008, pp. 33–40.

[21] Y. Ni, M. McVicar, R. Santos-Rodríguez, and T. De Bie, "An end-to-end machine learning system for harmonic analysis of music," *IEEE Audio, Speech, Lang. Process.*, vol. 20, no. 6, pp. 1771–1783, 2012.

[22] L. Oudre, Y. Grenier, and C. Févotte, "Template-based chord recognition: Influence of the chord types," in *Proc. ISMIR*, 2009, pp. 153–158.

[23] M. Mauch and S. Dixon, "Simultaneous estimation of chords and musical context from audio," *IEEE Audio, Speech, Lang. Process.*, vol. 18, no. 6, pp. 1280–1289, Jul. 2010.

[24] M. Khadkevich and M. Omologo, "Use of hidden Markov models and factored language models for automatic chord recognition," in *Proc. ISMIR*, 2009.

[25] R. Chen, W. Shen, A. Srinivasamurthy, and P. Chordia, "Chord recognition using duration-explicit hidden Markov models," in *Proc. ISMIR*, Porto, Portugal, Oct. 2012.

[26] J. Brown, "Calculation of a constant q spectral transform," *J. Acoust. Soc. Amer.*, vol. 89, pp. 425–434, 1991.

[27] E. Gómez Gutiérrez *et al.*, "Tonal description of music audio signals," Ph.D. dissertation, Univ. Pompeu Fabra, Barcelona, Spain, 2006.

[28] M. Khadkevich and M. Omologo, "Reassigned spectrum-based feature extraction for gmm-based automatic chord recognition," *EURASIP J. Audio, Speech, Music Process.*, vol. 2013, no. 1, pp. 1–12, 2013.

[29] M. Mauch, "Automatic chord transcription from audio using computational models of musical context," Ph.D. dissertation, Queen Mary Univ. of London, London, U.K., 2010.

[30] E. Gómez, "Tonal description of polyphonic audio for music content processing," *INFORMS J. Comput.*, vol. 18, no. 3, pp. 294–304, 2006.

[31] M. Mauch and S. Dixon, "Approximate note transcription for the improved identification of difficult chords," in *Proc. ISMIR*, 2010.

[32] M. Biasutti, "Sharp low-and high-frequency limits on musical chord recognition," *Hear. Res.*, vol. 105, no. 1, pp. 77–84, 1997.

[33] N. Ono, K. Miyamoto, J. Le Roux, H. Kameoka, and S. Sagayama, "Separation of a monaural audio signal into harmonic/percussive components by complementary diffusion on spectrogram," in *Proc. EUSIPCO*, 2008.

[34] M. Müller, S. Ewert, and S. Kreuzer, "Making chroma features more robust to timbre changes," in *Proc. ICASSP*, Taipei, Taiwan, Apr. 2009, pp. 1869–1872.

[35] M. Müller and S. Ewert, "Towards timbre-invariant audio features for harmony-based music," *IEEE Audio, Speech, Lang. Process.*, vol. 18, no. 3, pp. 649–662, Mar. 2010.

[36] B. C. Moore, *An introduction to the psychology of hearing*. Bingley, U.K.: Emerald Group, 2012, vol. 6.

[37] C. Roads, *The Comput. Music Tutorial*. Cambridge, MA, USA: MIT Press, 1996.

[38] T. Cho and J. Bello, "A feature smoothing method for chord recognition using recurrence plots," in *Proc. ISMIR*, Miami, FL, USA, 2011.

[39] T. Moon, "The expectation-maximization algorithm," *IEEE Signal Process. Mag.*, vol. 13, no. 6, p. 4760, Nov. 1996.

[40] K. Sumi, K. Itoyama, K. Yoshii, K. Komatani, T. Ogata, and H. Okuno, "Automatic chord recognition based on probabilistic integration of chord transition and bass pitch estimation," in *Proc. ISMIR*, 2008, pp. 39–44.

[41] P. Grosche and M. Müller, "Extracting predominant local pulse information from music recordings," *IEEE Audio, Speech, Lang. Process.*, vol. 19, no. 6, pp. 1688–1701, Aug.. 2011.

[42] M. Levine, *The jazz theory book*. Petaluna, CA, USA: Sher Music, 1995.

[43] S. Young, G. Evermann, M. Gales, D. Kershaw, G. Moore, J. Odell, D. Ollason, D. Povey, V. Valtchev, and P. Woodland, *The HTK book, v. 3.4*. Cambridge, U.K.: Cambridge Univ. Eng. Dept., 2006.

[44] K. Takeda, A. Ogawa, and F. Itakura, "Estimating entropy of a language from optimal word insertion penalty," in *Proc. ICSLP*, 1998.

[45] D. Gillick, L. Gillick, and S. Wegmann, "Don't multiply lightly: Quantifying problems with the acoustic model assumptions in speech recognition," in *Proc. IEEE Workshop Autom. Speech Recogn. Understand. (ASRU)*, 2011, pp. 71–76.

[46] K. Lee and M. Slaney, "A unified system for chord transcription and key extraction using hidden Markov models," in *Proc. ISMIR*, 2007.

[47] E. J. Humphrey and J. P. Bello, "Rethinking automatic chord recognition with convolutional neural networks," in *Proc. ICMLA*, Dec. 2012, pp. 357–362.

[48] T. Cho, K. Kim, and J. Bello, "A minimum frame error criterion for hidden Markov model training," in *Proc. ICMLA*, Dec. 2012, pp. 363–368.

Taemin Cho received the Ph.D. degree in Music Technology from New York University of New York, NY, U.S. He is a guitarist, composer, and music technologist. He received a B.S. in computer science from Inha University, Incheon, Korea, a B.M. with a double major in Performance (guitar) and Music Synthesis from Berklee College of Music of Boston, MA, U.S. and also received an M.M. in Music Technology from New York University. His main interests are real-time music controllers, real-time DSP, and Music Information Retrieval (MIR).

Juan P. Bello received the Ph.D. degree in electronic engineering from Queen Mary University of London, London, U.K. He was a Post-Doctoral Researcher and Technical Manager of the Centre for Digital Music, Queen Mary University of London. Since 2006, he has been an Assistant Professor of Music Technology at New York University, and a founding member of its Music and Audio Research Laboratory (MARL). He teaches and researches on the computer-based analysis of audio signals and its applications to music information retrieval, digital audio effects, and interactive music systems. He is a member of the IEEE and the society for music information retrieval (ISMIR), and a regular reviewer and contributor to digital signal processing and computer music journals and conferences. He is also a Researcher and member of the Scientific and Medical Advisory Board of Sourcetone, a music and health start-up.