EXPLORING COMMON VARIATIONS IN STATE OF THE ART CHORD RECOGNITION SYSTEMS

Taemin Cho, Ron J. Weiss and Juan P. Bello
Music and Audio Research Laboratory (MARL)
New York University, New York, USA
{tmc323,ronw,jpbello}@nyu.edu

ABSTRACT

Most automatic chord recognition systems follow a standard approach combining chroma feature extraction, filtering and pattern matching. However, despite much research, there is little understanding about the interaction between these different components, and the optimal parameterization of their variables. In this paper we perform a systematic evaluation including the most common variations in the literature. The goal is to gain insight into the potential and limitations of the standard approach, thus contributing to the identification of areas for future development in automatic chord recognition. In our study we find that filtering has a significant impact on performance, with self-transition penalties being the most important parameter; and that the benefits of using complex models are mostly, but not entirely, offset by an appropriate choice of filtering strategies.

1. INTRODUCTION

Chords are defined by the occurrence of harmonically related musical notes, either simultaneously or in quick succession. They are the smallest and most fundamental structures of the tonal system, which, it can be argued, makes them particularly adept at representing western popular music. Therefore, their identification is of great importance to a wide variety of applications in computer music, information retrieval and musicology. Chord transcriptions, however, are not readily available for most recorded music and can only be generated by highly-trained musicians. This motivates the development of automatic approaches to chord recognition, and explains the interest that this task has generated on the music computing community over the last decade.

A number of approaches to automatic chord recognition are discussed in the literature, several of which are mentioned in this paper. Notable amongst those, are the works of Fujishima [1], which introduced the use of chroma features to represent signal content, and of Sheh and Ellis [2], which pioneered the use of hidden Markov models (HMM) for chord recognition. It can be argued that all subsequent works are variations of the *standard* approach defined by both those papers, comprising a combination of chroma feature extraction, filtering and pattern matching, as shown in Figure 1. This is attested to by the homogeneity of most submissions to the yearly MIREX chord recognition task.

A direct consequence of this overlap is the existence of a number of important system variables that cut across approaches. Understanding these variables, their relation and relative importance in recognition results, is necessary in order to assess the limitations of the standard approach. Unfortunately, save a few exceptions (e.g. [3]), the literature lacks a comprehensive and holistic assessment on how parameter changes affect chord recognition.

The goal of this paper is to investigate the effect of common variables and to reveal their interrelationships, thereby providing valuable information that can guide future developments in automatic chord recognition. To this end, we perform a systematic evaluation including the most common and distinguishable variations in the literature. It must be clarified that covering the full range of possible variations is beyond the scope of this study. Instead we choose to use standard techniques for feature extraction and filtering, and concentrate our evaluation on variations of filtering parameters and the testing of different pattern matching approaches.

The remainder of this paper is organized as follows: Section 2 introduces the standard approach and discusses common variations at each stage; Section 3 presents the evaluation methodology and discusses the experimental results; while Section 4 includes our conclusions and directions for future work.

2. ARCHITECTURE OF A CHORD RECOGNITION SYSTEM

Most chord recognition systems share a common architecture comprised of four main stages: feature extraction, pre-filtering, pattern matching and post-filtering as seen in Figure 1. We discuss each of these steps in detail in the following sections.

2.1 Feature Extraction

The most popular features used for chord recognition are 12-dimensional Pitch Class Profile (PCP), or chroma features [1–12]. Chroma features represent the energy present in each of the twelve pitch classes, and are typically derived by

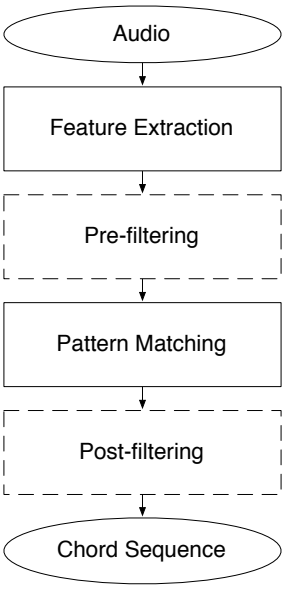
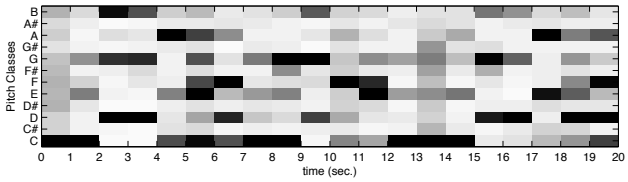
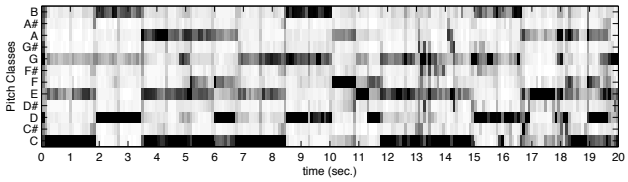


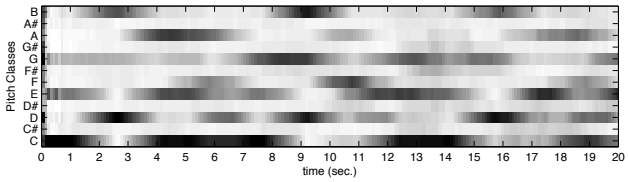
Figure 1. The basic architecture of the standard approach to automatic chord recognition (dotted lines indicate optional steps).



(a) 1 second hop size



(b) 96 ms hop size



(c) 96 ms hop size after smoothing with $N = 15$

Figure 2. Chromagrams computed from the first 20 seconds of "Let It Be".

mapping each frequency bin of the Discrete Fourier Transform (DFT) spectrum to a corresponding pitch class. Many variations on chroma feature extraction are documented in the literature [4, 8, 9, 13, 14].

The most common approach is based on the constant-Q transform, a spectral analysis technique in which the frequency channels are spaced logarithmically [15]. The constant-Q transform X_{cq} of an audio fragment $x(n)$ can be calculated as:

$$X_{cq}(k) = \sum_{n=0}^{L(k)-1} w(n,k)x(n)e^{-j2\pi f_k n} \qquad (1)$$

where k is the bin position, $w(n,k)$ is a window function of length $L(k)$, and f_k is the center frequency of the k^{th} filter bank. The calculation of the center frequency f_k is based on the frequencies of the equal tempered scale with:

$$f_k = 2^{k/\beta} f_{\min} \qquad (2)$$

where β is the number of bins per octave, and $f_{\min}$ is the minimum analysis frequency. From the constant-Q spectrum X_{cq}, the β-bin constant-Q chroma can be calculated as:

$$C_{cq}(b,n) = \sum_{m=0}^{M} |X_{cq}(b+m\beta,n)| \qquad (3)$$

where $b \in [1,\beta]$, and M is the total number of octaves in the constant-Q spectrum determined by the maximum analysis frequency $f_{\max}$. Finally the dimensionality of the β-bin chroma features computed in (3) is reduced to 12 bins by averaging adjacent bins using $\beta/12$-wide non-overlapping Gaussian windows.

The noteworthy thing in this stage is that most systems extract a chromagram with fixed frame rates (i.e. hop size) of 24 - 256 ms, which is significantly faster than the typical rate of chord changes in music. Only a few systems (e.g. [10]) use chroma features extracted from segmented audio frames with variable (and often slower) rates such as beat-synchronous chroma. Errors in the initial feature analysis, e.g. onset detection or beat tracking, in such systems propagate through the subsequent processing stages and can hurt overall performance.

In this paper we follow the fixed frame rate approach in order to focus on the effects of the remaining processing stages on overall performance. We use β = 36, with the analysis performed between f_{min} = 65.4Hz and f_{max} = 1046.5Hz [1] . The resulting window length and hop size are 8192 (186 ms) and 4096 (93 ms) samples respectively at 44100 Hz sample rate.

2.2 Pre-filtering

In order to precisely identify chord boundaries the frame rate of the chroma features must be faster than the rate of chord changes in a piece of music. This is demonstrated in Figure 2 (a) and (b) which compare chromagrams computed using 1 second and 96 ms hop sizes, respectively. The lower precision in Figure 2(a) is clearly visible. For example, the chord change at 11.5 seconds visible in Figure 2(b) is not present in Figure 2(a) at all. Moreover, the longer window exaggerates the influence of transient noise. For example, the short burst of noise in the A pitch class at 14 seconds in Figure 2(b), is drawn out over a full second in Figure 2(a).

However, the disadvantage of using such a short window is that the frames of the resulting chromagram are independent of the long term trend of the signal and respond to local changes, thus becoming sensitive to transients and noise in the signal. A popular technique to cope with this problem is to pre-process the chromagram prior to pattern matching using a low pass filter [1, 3–5, 8, 9, 11]. In this paper, we use

[1] In popular music, the harmonics of a musical note are usually stronger than the non-harmonic components up to 1 kHz [16].

a moving average filter which can be calculated as follows:

$$\hat{c}(n) = \frac{1}{N} \sum_{\tau=0}^{N-1} c\left(n + \tau - \frac{N-1}{2}\right) \quad (4)$$

where $c(n)$ is a frame of the chromagram, $\hat{c}(n)$ is a frame of the smoothed chromagram and n is the frame index. In this paper, $N = 0$ is defined as no filtering.

Intuitively, this technique improves pattern matching because it minimizes the effect of transients and noise in the signal by smoothing the features across neighboring frames. Figure 2(c) shows an example of the output of this process. The noise between 13 and 15 seconds in Figure 2(b) are filtered out in Figure 2(c).

2.3 Pattern Matching

The function of the pattern matching stage is to measure the fit of a set of predefined chord models, corresponding to each of the 24 major and minor triads, to each frame of the input chromagram, thus classifying each frame as being one of the 24 chords. The two most common approaches are based on a deterministic chord template generated by hand or a probabilistic chord model trained from examples of real music. The former approach is quite simple, but many variations of the probabilistic approach have been proposed. In this paper, we evaluate deterministic chord templates and three probabilistic chord models.

2.3.1 Binary chord template

A binary chord template is the simplest and one of the most popular chord models [1, 3–5, 7, 8]. This deterministic chord model is manually generated based on knowledge of the notes used in musical chords. In a binary chord template vector, each component corresponding to a chord-tone [2] is set to 1, and the other components are set to 0. [3] While some systems use variations of the binary chord template that incorporate information about higher harmonics produced by each chord-tone, recent studies have shown that simple binary chord templates are sufficient to obtain a good level of accuracy [7].

2.3.2 Probabilistic chord models

More sophisticated chord models are created by defining probability distributions for each chord class. A popular choice is the multivariate Gaussian distribution. In some systems, the Gaussian chord models are defined manually as with binary templates [3, 5]. More commonly, the distribution parameters are estimated from labeled data.

More precise chord models in the form of Gaussian mixture models (GMM) are sometimes constructed instead of single Gaussian models [12, 17]. Such models use multiple Gaussian distributions to represent each chord. Different components represent more nuanced instantiations of each chord in the training data, producing a more precise fit. This comes at the cost of requiring more sophisticated training using an Expectation-Maximization (EM) algorithm and increased computation when computing probabilities.

Finally, Khadkevich and Omologo [12] use a more complex chord model based on hidden Markov models (HMM) which enforces temporal continuity constraints not present in the other chord models. This chord model follows the topology typically used in automatic speech recognition, consisting of a 3-state, left-to-right HMM with the emission probability under each state consisting of a GMM.

All of the probabilistic models described in this section use Gaussian distributions with either diagonal or full covariance matrices. In most cases a diagonal covariance matrix is used under the assumption that the feature vector components are uncorrelated. In contrast, full covariance matrices capture correlations between different pitch classes. In this paper, we prepared two versions of probabilistic chord models, each with a diagonal and a full covariance matrices. For GMM chord models, we use 5, 10, 15, 20 and 25 mixture components.

The trained Gaussian and GMM models are denoted $M\ell_{\text{cv}}$, where ℓ specifies the number of components in the model and cv specifies the type of covariance matrix, diag or full; e.g. a single Gaussian model with diagonal covariances is referred to as $M1_{diag}$ and a 5 component GMM with full covariance is $M5_{full}$. The 3-state HMMs are denoted as $H\ell_{\text{cv}}$, and follow the same conventions as GMMs. For HMM models, ℓ indicates the number of mixture components used in each state of the model.

The parameters of the chord models are estimated from annotated training data using the EM algorithm. During training, the training data is segmented into 12 major triad and 12 minor triad segments based on the chord annotations. In order to compensate for the limited amount of training data, every chord segment is transposed to the key of C and this key-normalized data is used to train C-major and C-minor models. The trained chord models are then re-transposed to the remaining 11 major and 11 minor keys to define the remaining chord models.

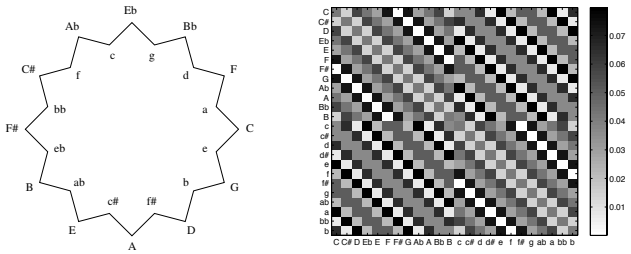
2.4 Post-filtering

The post-filtering stage shown in Figure 1 is used to smooth the sequence of predicted chord labels over time, thereby minimizing the number of spurious chords that only last for a small number of frames. Such mis-detections can be caused by short bursts of noise, which are very common in real music signals (e.g. see in Figure 2(b)). In most systems, post-filtering is performed with a Viterbi decoder, while some systems based on chord templates use a median filter instead [4, 7]. The Viterbi decoder finds the most likely sequence of chords based on the chord-type probabilities computed in the pattern matching stage.

The performance of the Viterbi decoding process is determined by the transition probability matrix, which describes the first-order temporal relationship between chords. Many researchers have concentrated on finding the optimal setting for these parameters. One approach has been to generate the matrix manually based on knowledge of music theory [3], while others have estimated the transition probabilities from music annotations [18].

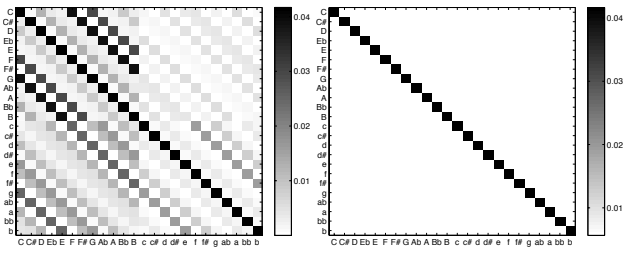
[2] The pitches which make up a chord are called chord-tones and any other pitches are called non-chord-tones.

[3] For example, the binary template for a "C Maj" 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 equally tempered scale from C.



(a) (b)

Figure 3. (a) Doubly-nested circle of fifths, (b) Chord transition probability matrix T_{C5}.



(a) (b)

Figure 4. (a) T_B and (b) T_U, both for 24 chord detection task after applying the transition penalty $P = 2.0$

Much of the previous work contains little discussion of the self-transition probability, which describes the probability of staying in the same chord from frame to frame. While some previous work reports improving the accuracy rate by manipulating the transition matrix, we argue that most of the improvement is contributed by a relatively high self-transition probability, which essentially acts to minimize the number of chord transitions. In a fast-frame-rate analysis, the probability of remaining in a chord is larger than that of moving to another chord, since the rate of chord changes is much slower than the frame rate. Thus, finding the optimal parameter for the self-transition probability tends to have more influence than the other parameters.

To evaluate this assumption, we define the transition penalty P, which is widely used in HMM-based speech recognition systems. This penalty adjusts the strength of the self-transition probability relative to transitions between different chords. It is applied as follows:

$$\log(\hat{a}_{ij}) = \begin{cases} \log(a_{ij}) - \log(P) & \text{for } i \neq j \\ \log(a_{ij}) & \text{for } i = j \end{cases} \quad (5)$$

where $A = [a_{ij}]$ is the original transition probability matrix and $\hat{A} = [\hat{a}_{ij}]$ is the modified matrix with penalty P.

In the case of the 3-state HMM model, this penalty is only applied to the transitions between chords, not transitions within the 3 states comprising the chord model. Since the internal HMM states already enforce temporal continuity within a chord (similar to self-transition probability), we set the diagonal entries of the chord transition matrix used for HMM post-filtering to zero. Therefore, P only changes the transition probabilities between different chord HMMs without touching the internal transitions between the 3 states.

We also evaluate the effect of different transition probability matrices on performance. As a baseline we define a uniform transition matrix T_U in which all transitions have the same probability (1/24 in our task).

For the transition probability matrix derived from music theory, we define a circle of fifths transition matrix T_{C5}, as proposed by [5]. In T_{C5}, the transition probability between two chords is derived from the distance between two chords in the doubly-nested circle of fifths (see Figure 3(a)). In this paper, for a fair evaluation of the parameter P (described in Section 2.4) against T_U, the diagonal entries of T_{C5} (self-transition probabilities) are adjusted to 1/24 and the other members in each row are normalized to sum to 23/24 (see Figure 3(b)).

Finally, we define a transition matrix estimated from training data, T_B. T_B, is estimated from bigrams of symbolic data in the training set. As in the training procedure for the chord models described in Section 2.3, we only calculate 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 C $\rightarrow$ Am and E $\rightarrow$ C#m are both counted as C $\rightarrow$ Am (**`I` $\rightarrow$ `vi`**).[4] The key-normalized bigrams are then transposed to the other major and minor roots to form the final matrix. We adjust the diagonal entries to 1/24 as we do for T_{C5}. Figure 4 shows an example of the transition matrices T_B and T_U after applying the transition penalty P using Eqn. (5).

3. EXPERIMENTS

In this section we describe a series of experiments to evaluate the effect of each processing stage on chord recognition performance. We evaluate the system variations using the well-known Beatles data set, 180 annotated songs from 12 Beatles albums (containing 13 discs).The ground truth chord annotations of the songs are kindly provided by C. Harte [19].[5] The evaluations are performed on 12 major, 12 minor and a no-chord detection task.

Each experiment is performed using a 13-fold cross validation. For each fold, one album is selected as a test set, and the remaining 12 albums are used for training. The chord recognition rate is calculated as follows:

$$\text{Accuracy} = \frac{\text{total duration of correct chords}}{\text{total duration of dataset}} \times 100\% \quad (6)$$

and is averaged across all cross-validation folds.

3.1 Pattern matching without filtering

In order to isolate the power of different pattern matching techniques from the effect of the other processing stages we evaluate the different chord models described in Section 2.3

[4] In this paper, Roman Numerals are open 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(b) or sharp(#) in front of a Roman Numeral lowers or raises the diatonic pitch by a half step. e.g. Both C $\rightarrow$ Am and E $\rightarrow$ C#m can be expressed as **`I` $\rightarrow$ `vi`**

[5] http://isophonics.net/content/reference-annotations-beatles

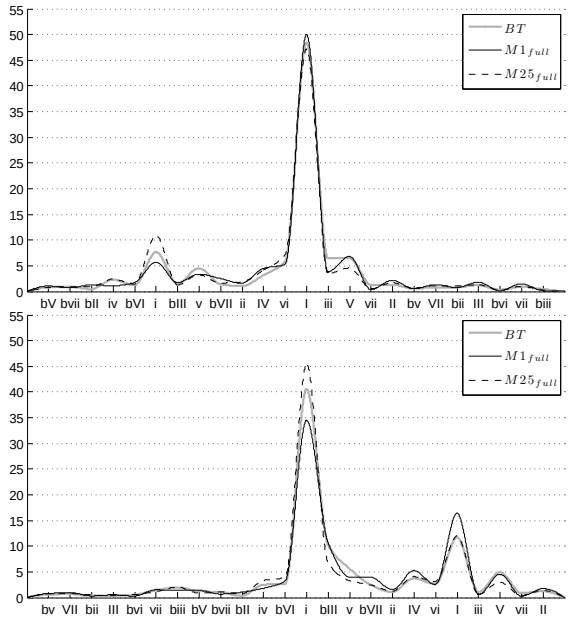


Figure 5. Chord classification errors from the experiment described in Section 3.1. The results are key-normalized to major (top) and minor (bottom) triads and averaged across all roots. The center labels (**`I`** of major triad detection and **`i`** of minor triad detection) represent correct classification and the remaining labels represent misclassified chords in Roman Numeral notation. The order of the labels follows the order of the doubly-nested circle of fifths in Figure 3(a).

without performing pre- or post-filtering. The 3-state chord HMM is excluded, because it requires post-filtering. The results are shown in Table 1.

Gauss. #		*1*	*5*	*10*	*15*	*20*	*25*	*BT*
M	*full*	46.8	45.0	44.7	45.1	**46.9**	46.8	46.69
	diag	40.1	42.1	42.6	43.6	45.5	45.5	

Table 1. Average accuracies of pattern matching methods with different chord models without filtering.

It would be reasonable to expect that performance should improve with increasing complexity of the chord model. However, the results in Table 1 contradict this expectation. In fact, none of the trained probabilistic models significantly outperform the simple binary template BT. All of the probabilistic models using diagonal covariance perform worse than BT, while those that utilize full covariance have performance roughly on par with BT. Another surprising result is that the performance of the GMM systems is no better than that of the single Gaussian chord model $M1_{full}$.

All of the chord models based on full covariance matrices perform better than the corresponding models based on diagonal covariance matrices. This gap is reduced by increasing the number of mixture components in the GMM. This trend is almost perfectly maintained across all experiments, therefore we only report results using full covariance matrices in the remaining experiments.

The distribution of chord detection errors is shown in Figure 5. The majority of errors are the result of confusions between harmonically related chords, i.e. those which share the same notes. For example, the chords in a parallel relationship (**`I`** and **`i`**), share a common root and fifth, and have only a semi-tone difference between the thirds. The figure also shows that actually $M25_{full}$ outperforms $M1_{full}$ and BT on minor chord detection. However, this is not reflected in the total overall results shown in Table 1, only 25% of the test set is comprised of minor chords, and so the decreased performance on major chord constributes more to the average performance.

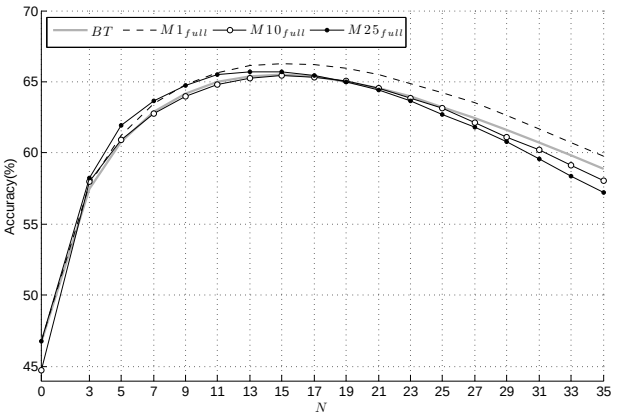


Figure 6. Average chord detection accuracy as a function of moving average pre-filter order N.

3.2 Effect of pre-filtering

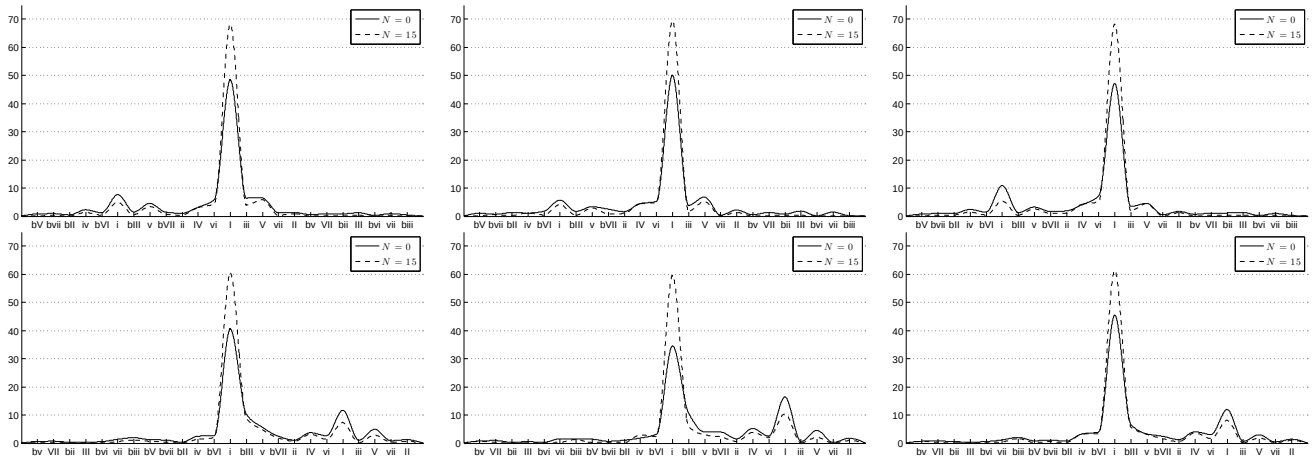
In this section we evaluate the effect of the pre-filtering procedure described in Section 2.2 on chord recognition performance. The chord models are retrained for each setting of the smoothing filter length N, and the filtered chromagrams are used for testing. The 3-state HMM template is excluded for the reasons described in Section 3.1.

The results are shown in Figure 6. Overall, pre-filtering improves performance over the results in Section 3.1 by about 20%. However, the best results for all chord models are almost the same (65.6% $\pm$ 0.1) except $M1_{full}$ (66.3%). The optimal N values for the chord models are also similar ($N = 15$). Once again, the number of mixture components in the GMMs has little effect on performance. The highest accuracy came from the simplest probabilistic model.

Many of the mis-classified frames in the previous experiment consist of very short ($1 - 5$ frame) segments caused by transient noise similar to that shown in Figure 2(b). The large improvement shown in this section is due to the fact that the pre-filtering process largely suppresses this noise.

It seems that training hurts the performance due to overfitting to the small training set and to reduced data variance caused by smoothing the features. In Figure 6, the accuracies of the 25 Gaussian models decrease faster than the single Gaussian models with increasing N. In other words the smoothed data tends to lead to overfitting, especially for chord models containing a large number of parameters.

Figure 7 shows the distribution of chord errors. As expected, a large portion of the performance improvement relative to the results reported in Section 3.1 is the result



(a) BT (b) $M1_{full}$ (c) $M25_{full}$

Figure 7. Major(top) and Minor(bottom) distribution of detections for different N

of fewer confusions between harmonically related chords. As a result, all the distributions of detections are similar to each other at the optimal N. The trained Gaussian model doesn't show any significant improvement against the binary template in this experiment.

3.3 Effect of post-filtering

In this experiment we evaluate the use of the different transition probability matrices defined in Section 2.4 and the effect of the transition penalty P. The pre-filtering parameter N is therefore fixed to 0 while P is varied. In order to smooth the output of BT, pseudo probabilities are calculated by taking the reciprocal of the Euclidean distances between chromagram frames and the chord templates. These are then passed through the Viterbi decoder.

Gauss. #		*1*	*5*	*10*	*15*	*20*	*25*	*BT*
T_U	P	15	23	22	20	19	15	1.25
	M	67.9	70.3	69.6	69.3	73.2	74.4	70.4
	P	16	18	14	11	8	9	
	H	67.4	71.7	72.5	72.7	74.4	75.0	
T_{C5}	P	15	22	22	20	19	16	1.5
	M	68.0	70.4	69.8	69.6	73.3	74.7	70.2
	P	16	18	14	10	9	8	
	H	67.6	71.9	72.6	72.9	74.5	75.0	
T_B	P	15	22	21	20	18	16	2.5
	M	68.5	70.7	70.3	70.2	73.8	75.1	70.6
	P	16	15	16	10	9	8	
	H	68.2	72.2	73.0	73.5	75.1	**75.6**	

Table 2. Accuracy of each chord model using different chord transition matrices. Each model-transition matrix combination is shown with the P value that maximizes accuracy.

The results are summarized in Table 2. For all systems, post-filtering shows a significant improvement over the results in the previous section, especially in the case of GMM chord models. Another surprising trend is that there is little difference in performance between the different transition matrices. The largest difference between them is only about 0.8%.

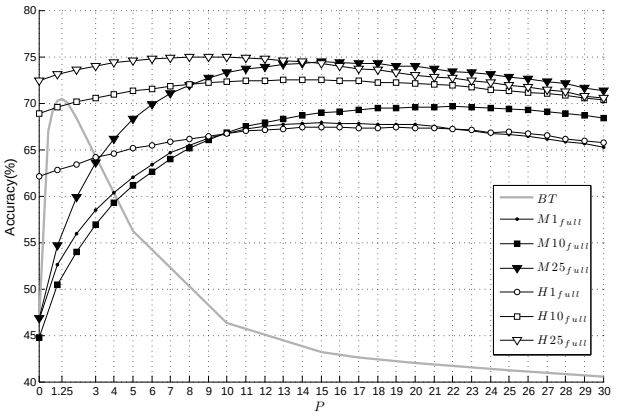


Figure 8. Average accuracy as a function of P using the T_U transition matrix ($N = 0$).

In addition, as shown in Figure 8, the parameter P has different effects on each of the chord models. The BT plot has a very steep curve and peaks at a smaller setting than any other model ($P = 1.25$). The system is overly sensitive to the transition penalty because the likelihoods of each chord class under this model tend to be very close together (i.e. it is not very discriminative), so, for large P, the transition probability overwhelms the likelihood.

On the contrary, the HMM chord model is much less sensitive to P than the other systems. This is due to the smoothing caused by the high internal self-transition probabilities already present within the chord models. The maximum accuracies of $H25_{full}$ and $M25_{full}$ are almost identical at each optimal P value. The two curves match up quite well if that of $M25_{full}$ is moved about 10 units to the left.

3.4 Combined pre- and post-filtering

Finally, we explore the relationship between pre- and post-filtering and evaluate all possible parameter combinations.

G. #	*1*	*5*	*10*	*15*	*20*	*25*	*BT*
N, P	7, 18	3, 23	3, 19	3, 18	3, 21	3, 17	3, 2.5
M	70.4	73.0	74.4	74.7	75.1	75.4	70.8
N, P	9, 7	3, 18	3, 13	3, 11	3, 11	3, 10	
H	70.2	73.7	74.9	74.9	75.4	**75.7**	

Table 3. The best accuracy of each chord model with T_B and optimal N and P.

Since, as described in Section 3.3, the exact transition matrix has a very small effect on performance, we only utilize T_B in this experiment.

The optimal N, P parameter combination for each system is shown in Table 3 along with the corresponding chord recognition accuracy. The best results occur using relatively little pre-filtering ($N = 3$) with transition penalty P similar to the optimal settings found in the previous experiment. The combination of both filtering stages has minimal effect on the best performing system, however it does bring the performance of the systems based on simpler GMM chord models to the same level as the HMM systems. From this we can conclude that the additional pre-filtering stage can be used in place of the more computationally complex HMM chord model without significantly affecting performance.

Compared to the results of post-filtering alone in Table 2, pre-filtering increases the accuracy of all chord models. The smoothing across very short frames ($N = 3$) yields significantly improved accuracy, especially in the case of GMM chord models. However, this effect decreases as number of Gaussians increases. For $M25_{full}$ the smoothing has no significant effect on performance. Overall, the combination of pre- and post-filtering decreases the difference in performance between the GMM chord models. More than doubling the size of the model from 10 to 25 mixture components increases performance by only 1%.

A similar trend can be seen in the performance of the HMM systems. More notably, the small gaps (less than 1%) between GMM and HMM systems with the same number of Gaussians implies that additional pre-filtering can compensate for the additional smoothing present in the more complex model. The 3-state left-to-right HMM architecture requires that each recognized chord have a minimum length of 3 frames. As described in the previous section, this has an implicit smoothing effect and provides better time-persistence than a single frame level chord detection. The results in Table 3 demonstrate that the overall effect is similar to that of the 3 frame moving average filter.

3.5 Summary

Table 4 summarizes the experiments described in this section. All combinations of filtering strategies and pattern matching techniques are shown. T-tests show that differences in accuracy greater than 1% are statistically significant ($p < 0.01$). The performance improvements seen when moving down each column demonstrate the very large impact of filtering on the accuracy of chord recognition system. Post-filtering has the largest impact, primarily due to the self-transition penalties, and the combination of pre- and post-filtering leads to relatively small improvement over the optimal choice of post-filter alone, however this improvement was only statistically significant for $M1$ and $H1$. Despite the great deal of research investigating different chord models, we have observed relatively small performance differences between the models in our experiments. Given optimal filtering settings, all systems perform within 5% of each other (bottom row of Table 4). It is worth noting that the chord models used most often in the literature are based on diagonal covariance matrices [2, 6, 12, 17, 18]. In our experiments, diagonal covariance models performed an average of about 2% lower than the corresponding models based on full covariance matrices.

Chord Models	BT	$M1$	$M25$	3-state HMM	
				$H1$	$H25$
No Filtering	46.69	46.76	46.77		
Pre-filtering	65.50	66.27	65.72		
Post-filtering	70.60	68.52	75.14	68.16	75.56
Pre & Post	70.82	70.44	75.40	70.23	**75.70**

Table 4. The best results of each chord model for each experiment.

There is a general trend of improving performance with increasingly complex chord models, but the effect is dominated by the number of parameters used by each model, i.e. the number of mixture components or states. The best results are obtained using the most complex system based on HMM chord models. The system contains the largest number of parameters and therefore has the highest risk of overfitting to the small training data set used in these experiments.

4. CONCLUSION

This paper presents a systematic evaluation of increasingly complex variations of the standard approach to automatic chord recognition, with a focus on the impact of filtering and pattern matching strategies. Experimental results show that filtering, both before and after pattern matching, has a significant impact on the accuracy of recognition. In the case of pre-filtering we found that variations of the parameter N have a similar effect across different models, with the optimal value increasing performance by as much as 20% upon the unfiltered case. Optimal post-filtering can increase performance by little less than 30%, with high self-transition probabilities (determined by the parameter P) being entirely responsible for this change. Unlike N, optimal values of P have to be carefully chosen for each different model. Combining pre- and post-filtering brings about only marginal improvement over the optimal choice of post-filtering. Surprisingly, we found that the effect of different transition matrices is negligible. This indicates that, at fast frame-rates, any attempts to encode information about likely chord transitions is rendered moot by the need to enforce continuity in the estimations.

While the best overall results are obtained for $M25_{full}$ and $H25_{full}$, it is worth noting that the benefits of using

complex models are mostly offset by an appropriate choice of filtering strategies. Notably, the best performance using simple binary template matching is only 5% less than the best performance using a network of 3-state HMMs, with mixtures of 25 full-covariance Gaussians per state. This is troublesome not only because of the significant difference in computational cost and overall complexity between these two models, but also because the extensive testing and parameter selection on such a small and homogeneous dataset most likely means that the difference is attributable to overfitting and is not generalizable to other music.

Of course, these findings are only valid for fast, fixed-rate features, and could be alleviated by slowing down the feature rate, or by using variable-rate methods such as beat segmentation. However, the former will have an impact on accurate detection of chord transition boundaries (which, if increases in accuracy are forthcoming, might be less of an issue), while in the case of the latter, it is far from clear that the current state of the art in beat tracking can ensure robust performance across a wide variety of music, and thus avoid issues of error propagation.

In either case, it is worth pointing out that lower feature frame-rates will considerably reduce the amount of available data, which will in turn negatively affect our ability to train complex models, such as the ones considered in this paper. This, together with the above mentioned issues of overfitting (which we believe to be widespread in chord recognition research), highlights the need for data collection as a necessary step in the development of better approaches. More data will also support the use of discriminative models for pattern matching (which has proven successful in MIREX-09 [20]) and for supervised training of complex dynamic models integrating rhythmic, harmonic and structural analysis [10]. Finally, we have yet to evaluate the impact of different feature extraction strategies and test new methods (e.g. [14]) that have already shown promise in related music analysis tasks.

5. ACKNOWLEDGMENTS

This material is based upon work supported by NSF grant IIS-0844654 and by IMLS grant LG-06-08-0073-08.

6. REFERENCES

[1] T. Fujishima, "Realtime chord recognition of musical sound: a system using common lisp music," in *Proc. ICMC*, pp. 464–467, 1999.

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

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

[4] C. Harte and M. Sandler, "Automatic chord identification using a quantised chromagram," in *Proc. of the Audio Engineering Society (AES)*, 2005.

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

[6] M. Ryynänen and A. Klapuri, "Automatic transcription of melody, bass line, and chords in polyphonic music," *Computer Music Journal*, vol. 32, no. 3, 2008.

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

[8] K. Lee, "Automatic chord recognition using enhanced pitch class profile," in *Proc. ICMC*, 2006.

[9] M. Khadkevich and M. Omologo, "Phase-change based tuning for automatic chord recognition," in *Proc. of Digital Audio Effects Conference (DAFx)*, 2009.

[10] M. Mauch, K. C. Noland, and S. Dixon, "Using musical structure to enhance automatic chord transcription," in *Proc. ISMIR*, pp. 231–236, 2009.

[11] H. Papadopoulos and G. Peeters, "Simultaneous estimation of chord progression and downbeats from an audio file," in *Proc. ICASSP*, pp. 121–124, 2008.

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

[13] M. Stein, B. M. Schubert, M. Gruhne, G. Gatzsche, and M. Mehnert, "Evaluation and comparison of audio chroma feature extraction methods," in *Proc. of the Audio Engineering Society (AES)*, 2009.

[14] M. Müller, S. Ewert, and S. Kreuzer, "Making chroma features more robust to timbre changes," in *Proc. ICASSP*, vol. 0, pp. 1877–1880, 2009.

[15] J. Brown, "Calculation of a constant Q spectral transform," *Journal of the Acoustical Society of America*, vol. 89, no. 1, pp. 425–434, 1991.

[16] N. C. Maddage, "Automatic structure detection for popular music," *IEEE Multimedia*, vol. 13, pp. 65–77, 2006.

[17] J. Reed, Y. Ueda, S. M. Siniscalchi, Y. Uchiyama, S. Sagayama, and C.-H. Lee, "Minimum classification error training to improve isolated chord recognition," in *Proc. ISMIR*, pp. 609–614, 2009.

[18] K. Lee and M. Slaney, "Acoustic chord transcription and key extraction from audio using key-dependent hmms trained on synthesized audio," *IEEE Transactions on Audio, Speech and Language Processing*, vol. 16, no. 2, pp. 291–301, 2008.

[19] C. Harte, M. Sandler, S. Abdallah, and E. Gómez, "Symbolic representation of musical chords: a proposed syntax for text annotations.," in *Proc. ISMIR*, 2005.

[20] A. Weller, D. Ellis, and T. Jebara, "Structured prediction models for chord transcription of music audio," in *Proc. ICMLA*, vol. 0, pp. 590–595, 2009.