Automatic Chord Recognition from Audio

Alex R. Emmons
Division of Science and Mathematics
University of Minnesota, Morris
Morris, Minnesota, USA 56267
emmon046@morris.umn.edu

ABSTRACT

Automatic chord recognition from audio is used in the area of *Music Information Retrieval* (MIR) to document and categorize music. In addition to providing harmony to music, chords also provide a way to describe the harmony of a piece. In almost every chord recognition system the audio signal is represented by a *Pitch Class Profile* (PCP), which measures the intensity of energy in each of the frequency regions where musical notes occur [5]. Some systems perform what is known as preprocessing before generating a PCP, to get rid of unwanted frequencies in the audio file. The next step, known as pattern matching, is to assign chord labels by matching the harmonic features to a set of chord models. In this paper we will discuss these processes in greater detail and compare the results of three research cases, each of which uses a different chord recognition system.

Keywords

Automatic chord recognition, hidden Markov models, pitch class profile, signal processing

1. INTRODUCTION

A chord is a set of tones played simultaneously. A chord progression is a sequence of chords over time and is what describes the harmony of a piece [2]. Automatic chord recognition is the process of extracting a chord progression from an audio file. These chord sequences are used by musicians as lead sheets (summaries containing chords, melody, and lyrics) as well as by researchers for tasks such as key detection, genre classification, and lyric interpretation. Performing chord analysis by hand is time consuming, prone to human error, and requires two or more trained experts. This is what makes automatic chord recognition an important area of research [4].

The two main steps of automatic chord recognition are feature extraction and pattern matching. Feature extraction is the process of extracting useful information from audio files, and pattern matching is how chord labels are applied to that data.

There are many challenges encountered by systems that process audio signals. There are background noises, percussion instruments, and other unwanted tones in audio recordings. It is also difficult to distinguish when chords change and to line these points up exactly with the beat. Preprocessing helps eliminate unwanted information from the audio files before or during the feature extraction step, depending on the system. An overview of the chord recognition system used in [5] can be seen in Figure 1.

These systems have been improving and becoming more usable in recent years. This paper will compare three different systems that use a variety of techniques in each step of the process. By looking at the components of the highest performing systems, we will determine the most effective methods used in each step.

2. BACKGROUND

In order to explain the process of automatic chord recognition, some general information about feature extraction and pattern matching is needed.

2.1 Feature Extraction

The first step of generating a chord progression from audio data is processing the signal to extract harmonic features. Feature extraction is a fairly simple process, but can become more complex due to the addition of optimization steps to increase accuracy [4]. Preprocessing is one of these steps, performed during feature extraction, which results in a *Pitch Class Profile* (PCP), representing what notes are present over time.

2.1.1 *Preprocessing*

In Figure 2, the light areas show where frequencies have been detected, and the dark areas show empty space. It is clear that frequencies other than just the chord tones have been detected because the light areas are not solid white and the dark areas not solid black. The goal of preprocessing is to reduce as much of this background noise as possible from the audio file, in an effort to provide a smooth and clear PCP. Another issue is that musical instruments produce a series of harmonics at higher and lower frequencies than the tone that is played. These tones, called overtones, can confuse feature extraction techniques, so they need to be removed. These two issues, background noise and overtones, are usually addressed separately during preprocessing.

UMM CSci Senior Seminar Conference, December 2014 Morris, MN.

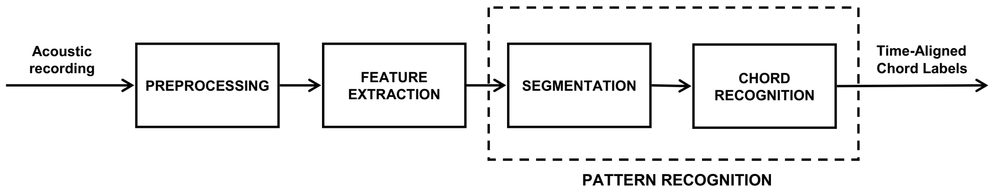

Figure 1: Overview of the chord recognition system used in research case 1 [5].

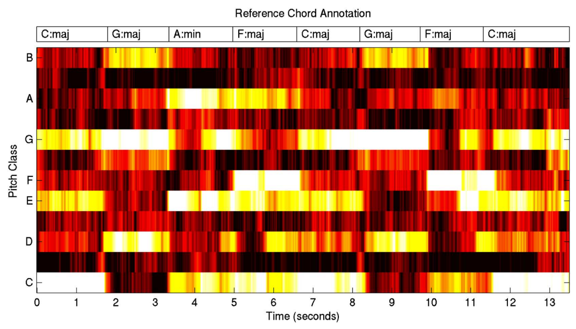

Figure 2: A typical chromagram, or PCP, generated from the opening to *Let It Be* (Lennon/McCartney). Pitch class (chroma) at time t is shown by the lightness at that point. The true chord progression (simplified) is shown above for comparison.

2.1.2 *Pitch Class Profile*

The pitch of a note is measured in two dimensions - height and chroma (pitch class). Height tells which octave a note belongs to, and chroma tells where a note stands within the octave (the name of the note). Height is not a factor in determining chord type because two notes that are an octave apart have the same chroma value. A chromagram, or PCP, is a 12-dimensional vector representation of chroma, representing the intensity of each of the twelve semitones in the chromatic scale, over a period of time. An example of a common PCP, along with the actual progression, can be seen in Figure 2 [4].

For over a decade PCP has been the most popular way to represent harmonic features for chord recognition. Most new approaches are variations or refinements of this approach [1]. These systems can have many more steps in converting audio to PCP including tuning correction, which compensates for music that is not tuned to standard pitch $A4 = 440\ Hz$, and beat-synchronization, which calculates the average pitch between beats to get rid of changes caused by noise and other transients. [4].

2.2 Pattern Matching

Almost all chord recognition systems use PCP or some other chroma-based feature extraction technique. What differentiates these systems is the mechanism used to assign the chord labels. Generating the chord model against which the PCP will be matched can be done either by hand (using musical knowledge), or stochastically (by deriving it from real-world music). The most basic way to model chords is by hand, describing the known note distribution of a chord. An example of a binary chord template (individal chord model) for a C major *triad* (chord containing 3 notes) would look like [1 0 0 0 1 0 0 1 0 0 0 0], where each digit left to right follows the chromatic scale starting at C. A 1 indicates the presence of a note and 0 represents the absence of a note. Using this method, each chord template is created and then tested on each *frame* (time point around 25ms, depending on the system) of a PCP, to find the best fit. The problem with this method is that a lot of times chord tones are not all played at the same time or held for the same duration. More sophisticated chord models are generated stochastically, by defining probability distributions for each chord type that a system can recognize [1].

2.2.1 *Hidden Markov Models*

A *hidden Markov model* (HMM) is a statistical model which describes a finite set of states, in this case chords. Since we do not know the chord progression for a given audio file, these states are considered to be hidden, the observed states are the PCP frames. Transitions between these states are given probabilities that describe the likelihood of transitioning from one state (chord) to another. In the research cases reviewed in this paper, *supervised* HMMs are used, meaning that these probabilities are *learned* from a set of labeled training data (audio that has been assigned chord labels by hand or some other system) [6]. The chord transition probabilities are learned for a chord by dividing the number of transitions to each chord by the total number of transitions from that chord. This is done for each chord in the training data. An example of transition probabilities from the C major chord can be seen in Figure 3. The Viterbi algorithm is then used, which estimates the the best previous chord for each frame recursively, storing the paths in a table. Using the observed states and known transition probabilities, the most likely sequence is estimated and unlikely transitions are restricted [1].

2.2.2 *Gaussian Mixture Models*

In a *Gaussian mixture model* (GMM), more detailed models for each chord type are created by averaging multiple PCPs for that type. This is done for each chord type for each of the 12 notes. Multiple Gaussian *components* are also used, which represent different variations of the same chord (i.e., chords with notes excluded or doubled up), providing a more accurate fit. The fit of a frame is measured by the likelihood of a model for that frame.

For GMM training, PCPs from the training data are segmented by chord type (i.e., 12 major, 12 minor, depending on what chords are being recognized). To increase the number of training samples, all PCP values for these segments are rotated so that the *root* (name) is C. These segments are

	Type	Sample Rate	FFT Length	Liftering	HPS Ratio
FV1	FB	44100	32768	yes	5
FV2	PCP	11025	4096	no	1
FV3	PCP	44100	32768	no	1
FV4	PCP	44100	32768	yes	5

Table 1: Feature Vectors, or combinations of methods used for feature extraction, for isolated chord recognition in research case 1 [5].

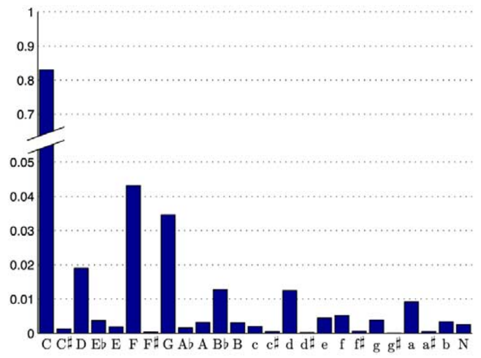

Figure 3: An example of transition probabilities from the C major triad, learned from training data.

then averaged to create models for C-major and C-minor chords, which are then copied and shifted for the remaining 11 keys [4].

2.2.3 Support Vector Machines

Support Vector Machines (SVMs) are another type of supervised learning model used for pattern matching. Given a set of labeled training data, SVMs assign labels to segments of the test data. A full explanation of SVMs is outside the scope of this paper, as they are only used in one research case for comparison. Many systems use existing tools for SVM classification.

3. RESEARCH CASES

This paper looks at three research cases that involve automatic chord recognition. All of these cases include feature extraction and pattern matching. Here we will give an overview of each system, the datasets that were used, and a summary of the results.

3.1 Case 1: Effects of Proper Signal Processing

The first research case [5] uses a chord recognition system that begins with a preprocessing block, followed by feature extraction, where the PCP is calculated. After this, the signal is segmented along predicted chord boundaries, and then chord labels are assigned to each segment (pattern matching). An overview of this system can been seen in Figure 1.

There are two stages in the preprocessing block, to address background noise and overtones. The first step, Homomorphic Liftering, is a method of separating out the frequencies of musical tones from background and system noise. This is done by finding strong frequency peaks in areas corresponding to the pitch range of the notes. Frequencies above and below a specified range can also be removed to reduce noise. The second step, known as the *Harmonic Product Spectrum* (HPS), is a method which emphasizes frequencies when their overtones are present. HPS is calculated by compressing the spectrum by factors of 1 to R (i.e., 1:2 compression ratio) and multiplying the resulting compressed spectra. The resulting output energy is then summed according to pitch class before a PCP is created.

The first step of pattern matching for this system is chord segmentation, where the audio signal is segmented at the boundaries where chords change. This can be difficult when the notes of a chord are not played all at once or not held for the same length. To find the points where change has occurred the PCP is analyzed frame-by-frame to find significant change in pitch-class content.

The next and final step in this system is assigning chord labels to the segments. Given an instance of the PCP for a region of the audio, the most probable chord label is picked from a set of training PCPs. In this research case the use of GMMs and SVMs are compared for classification. For SVM classification, a tool called the OSU SVM Classifier MATLAB Toolbox was used (as explained in [5]).

3.1.1 Datasets

In this case *Musical Instrument Digital Interface* (MIDI) data was used to create the audio. Two datasets were used: one of isolated chords synthesized on piano and strings, and one of continuous single-instrument music synthesized on piano.

The isolated chord dataset consisted of 7790 chords and inversions (where the notes are stacked in a different order). Three chord complexity levels were tested, labeled DS1, DS2, and DS3 (seen in Table 2). DS1 involved the four common triad types (major, minor, augmented, and diminished), across all 12 notes. No other chords were used in DS1. DS2 added variations of the *7th chord* (chords containing four notes): major 7th, minor 7th, dominant 7th, fully diminished 7th, and half diminished 7th. In DS3 all 11 different chord types that the system could recognize were used.

Four *Feature Vectors* (FV), or combinations of methods used for feature extraction, were compared on each of these complexity levels, seen in Table 1. FV1 started with preprocessing using homomorphic liftering with a lower cutoff of 30Hz and upper frequency cutoff of 4kHz, then computing HPS with a compression ratio R = 5. Instead of PCP, FV1 uses an 84 dimensional vector (representing 7 octaves with 12 notes per octave). The hypothesis was that this would provide more information because the octaves weren't flattened down to one. FV2 had a lower sampling rate (audio quality), shorter *Fast Fourier Transform* (FFT) window length (used in converting audio to frequency domain), and no preprocessing was performed on the audio signal. This feature vector was chosen to match a previous system that

	Label given in:		
Chord Label	**DS1**	**DS2**	**DS3**
Major	Major	Major	Major
Minor	Minor	Minor	Minor
Major 7	-	Major	Major 7
Minor 7	-	Minor	Minor 7
Dom. 7	-	Major	Dom. 7
Dim.	Dim.	Dim.	Dim.
Full Dim.	-	Dim.	Full Dim.
Half Dim.	-	Dim.	Half Dim.
Augmented	Aug.	Aug.	Augmented
Sus. 4	-	-	Sus. 4
7 Sus. 4	-	-	7 Sus. 4

Table 2: Chord complexity levels used in research case 1 [5].

was being comparing against. FV3 increases the sample rate and FFT, but still leaves out preprocessing. FV4 introduces preprocessing using the same homomorphic liftering and HPS as FV1.

The continuous single-instrument audio dataset consisted of 50 hymn verses selected from the Trinity Hymnal, a MIDI collection of 761 hymns, synthesized on piano. 40 were used for training and 10 for testing. Labels similar to DS3 were used, with the exception of 7 Sus. 4, Full Dim., and Half Dim. chords, as they were not common in the dataset. FV4 was used as the feature vector [5].

3.1.2 Results

For the isolated chords dataset, this system was trained using chords synthesized on a piano, and was tested on chords synthesized on both piano and on strings. The dataset was randomly divided with 80% of the chords used for training and 20% for testing. Five of these training and testing sets were created and the recognition rates from these were averaged. The overall chord recognition accuracies for GMMs can be seen in Table 3 for piano, and in Table 4 for strings. FV4 performed the best with the chords played on strings, and showed the least difference between recognizing piano and string chords [5]. The results of classification using SVMs can be seen in Table 5. Using SVMs clearly outperformed GMMs when tested with the instrument that they were trained with, yielding accuracy rates up to 95%. This was not the case however, when SVMs were tested on strings when trained on piano. In this case SVMs actually did not work because an SVM requires knowledge of the type of data in the testing set, so these results were not included.

For the continuous single-instrument dataset, the number of *scatter points* (7 scatter points means the partition frame plus 3 frames on each side) were compared, as shown in Table 6. Recognition accuracy of around 88% was achieved. Melody tones that were not part of the chord made this more difficult than labeling the isolated chords [5].

3.2 Case 2: HMM Trained with Audio from Symbolic Data

In the next research case [2], symbolic data in Humdrum data format is used to generate training data for an HMM. Humdrum is a software toolkit used for music research. The data is used to generate a chord label file (containing chord names and times), and an audio file (synthesized from MIDI),

Feature Vector	DS1	DS2	DS3
FV1	83.68	61.85	57.24
FV2	90.33	82.44	82.26
FV3	**91.76**	84.20	84.09
FV4	85.64	79.40	78.93

Table 3: Isolated chord recognition accuracy using GMM, training set: piano, testing set: piano [5].

Feature Vector	DS1	DS2	DS3
FV1	68.06	42.00	33.62
FV2	42.72	18.60	16.30
FV3	43.49	22.00	18.31
FV4	**86.94**	80.23	80.18

Table 4: Isolated chord recognition accuracy using GMM, training set: piano, testing set: strings [5].

from the same data. Chroma analysis is then performed on the audio file to get a PCP, which is used as input to the trained HMM along with the chord label file. This system is visualized in Figure 4.

A 36-state HMM is used, with each state representing a chord (major, minor, and diminished for each 12 pitches). Once the model parameters are learned, the Viterbi algorithm is applied to find the optimal sequence in a maximum likelihood sense.

3.2.1 Datasets

Two training datasets were used: the first consisted of 81 solo piano pieces by J.S. Bach, Beethoven, and Mozart, and the second consisted of 196 string quartet pieces by Beethoven, Haydn, and Mozart. For the testing datasets, five piano solos and five string quartet pieces were selected from the Kostka and Payne's book, which includes hand-marked analysis and audio recordings done by the composers. Both of these sets were tested using both of the training datasets, as well as third that consisted of the first two combined, resulting in six possible training - testing pairs. The output of the system was compared to the hand-marked data to check for accuracy [2].

3.2.2 Results

The results of the experiments in this case can be seen in Table 7. The recognition rate was highest for the combined training dataset, at 80%, although the difference is not significant. Further analysis on the results showed that the highest error came from non-chord tones in the melody, especially in faster pieces. This is because the analysis window would span over multiple chord changes, which confused the system. Other issues were caused by the system treating 7th chords the same as triads because 7th chords contain two triads.

3.3 Case 3: Importance of Individual Components

The final research case [1] consists of four experiments, testing different methods used for each step of the process. The overall recognition system in this case is defined as a feature extraction step and pattern matching step, with optional pre-filtering and post-filtering steps. In this case preprocessing is part of the feature extraction step. An overview

Expt.	Highest Accuracy	Pre-filtering	Pattern Matching	Post-filtering
1	58.30	-	1 Gaussian component	-
2	71.22	Moving average filters	1 Gaussian component	-
3	**77.90**	-	25 Gaussian components	HMM
4	77.58	Moving average filters	25 Gaussian components	HMM

Table 8: Results from research case 3 [1], showing the highest accuracy in each experiment, and the components used to achieve it.

Feature Vector	DS1	DS2	DS3
FV1	93.43	88.21	86.52
FV2	94.78	93.26	93.13
FV3	**95.23**	94.31	94.24
FV4	90.56	88.08	87.74

Table 5: Isolated chord recognition accuracy using SVM, training set: piano, testing set: piano [5].

Number of Scatter Points			
3	**5**	**7**	**9**
72.73	87.77	88.07	**88.42**

Table 6: Continuous single-instrument recognition accuracy with varying number of scatter points [5].

of the system can been seen in Figure 5. Pre-filtering is applied to PCPs directly before pattern matching, and consists of averaging frames between beats, yielding a beat-synchronized PCP. Post-filtering is performed after pattern matching, in the form of an HMM using a Viterbi decoder.

In the first experiment, different combinations of chroma features and chord models were tested to show whether or not model complexity affects performance of the system. The second experiment looked at the effect of different pre-filtering techniques, while the third looked at the effect of post-filtering. The fourth experiment used both pre-filtering and post-filtering, in all possible parameter combinations.

3.3.1 Datasets

Each of the four experiments in this research case were performed on 180 Beatles songs, 20 Queen songs, 100 songs from the RWC (Real World Computing) pop dataset, and 195 songs from the US-Pop dataset, for a total of 495 songs. The RWC and US-Pop dataset were assigned chord labels by hand, while the Beatles and Queen chord models were generated stochastically. For training, 5-fold cross validation was used with each group having 99 songs selected randomly. For each fold one group is selected and the other four are used for training. Accuracy is represented by the total duration of correct chords out of the total duration of the dataset [1].

3.3.2 Results

The results of the four experiments conducted in this case can be seen in Table 8. In all four experiments the best results were found using a combination of these two preprocessing methods: de-emphasizing high and low frequencies and using log compression, which limits the dynamic range caused by multiple instruments at different volumes. Each experiment was also tried using 1, 5, 10, and 25 Gaussian components, to test if a more complex chord model would increase accuracy. For each experiment the best result and components used to achieve it are shown in Table 8.

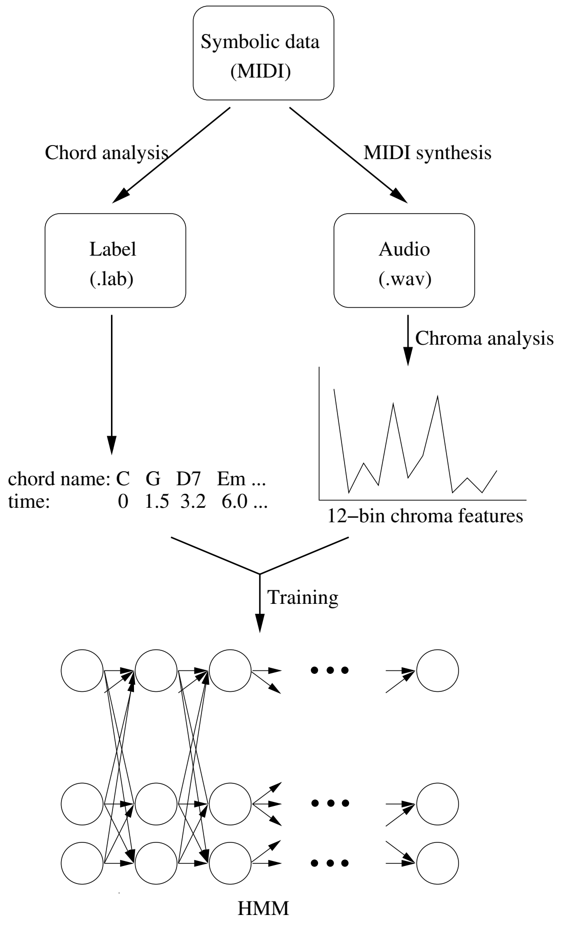

Figure 4: Overview of the HMM trained with audio from symbolic data in research case 2 [2].

In the first experiment, no filtering was applied during the process. This experiment was testing the difference between preprocessing techniques in the feature extraction stage (the best combination is mentioned above). We would expect to see higher accuracy with more Gaussian components used, but that was not the case in this experiment, where the best score was found by using only one component. The highest error was in distinguishing major and minor chords with the same root, because two of the notes are shared and the third is only a half-step different. For the second experiment, pre-filtering was applied using *moving average filters*, which look for noisy frames and smooth them across the neighboring frames. The highest accuracy in this experiment was again achieved by using only one Gaussian component. The third experiment used post-filtering with

Training Data	Test Data	Recognition Rate
Piano	Piano	68.69
String Quartet	Piano	73.40
Piano & Strings	Piano	74.41
Piano	String Quartet	79.35
String Quartet	String Quartet	79.76
Piano & Strings	String Quartet	**80.16**

Table 7: Recognition results for all six possible training - test pairs in research case 2 [2].

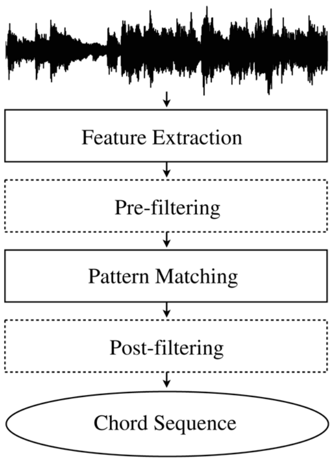

Figure 5: General layout of the chord recognition system used in research case 3 [1]. In this case preprocessing is included in the feature extraction step.

no pre-filtering, and here we see that the best performance was achieved by using more Gaussian components. This is because post-filtering is dependent on the probability values for different chord models. The final experiment, using both pre-filtering and post-filtering, showed about the same result as post-filtering alone. This is because moving average filters sometimes did too much smoothing and ended up blurring out chord boundaries and other signal detail.

4. CONCLUSIONS

The highest accuracy that we see in this paper was around 95% for the isolated chord dataset using SVMs in [5]. This is more of a proof of concept since these systems are aimed at providing chord data for an entire song. If we look at the results of all experiments tested on continuous music, we see that the highest accuracy achieved was around 88% for the continuous music dataset in [5]. The components used in this experiment were: preprocessing using homomorphic lifering and HPS with a ratio of R = 5, chord segmentation using SVMs, and chord recognition using SVMs. This was using a relatively small dataset, they mention in [5] that the training procedure for SVMs is very complex with large amounts of data. This implies that the system used for chord recognition is somewhat dependent on the dataset that is being tested. Systems can be tailored specifically to the type of instrument and chords that are present in the dataset.

There are still many issues and pitfalls with these and all chord recognition systems. Music that has a lot of different instruments, fast chord changes, and types of chords that are not recognized can cause significant problems if they are not expected. The pattern matching techniques in this paper assume that individual chords are independent of each other, but in real-world music some chords can only be labeled correctly by the context of the surrounding chords [1]. An example is a secondary chord, which is borrowed from another key. These chords would be given a chord label that is not in the key of the song, rather than the secondary label that would be applied by hand. Detecting these and other non-regular chords would be very difficult for these systems, as they only recognize the chords that they are designed for.

4.1 Future Work

With the increasing amount of data that is available online, some systems have been developed that scrape online databases of songs that have already been labeled to use for training data. The advantage here is having a large amount of labeled chord data without having to generate it. The system in [3] was able to scrape chord information from `e-chords.com` for over 75,000 songs. There are also genre-specific models, which account for different chord transition probabilities in different genres of music. These models can also be used to identify genre by testing with all of the genre models and selecting the one with the best result [4]. Progress is being made in this area, including mobile apps that can provide chord data for audio captured with the microphone.

5. ACKNOWLEDGEMENTS

Many thanks to Elena Machkasova, KK Lamberty, and Scott Steffes for their invaluable feedback.

6. REFERENCES

[1] T. Cho and J. Bello. On the relative importance of individual components of chord recognition systems. *Audio, Speech, and Language Processing, IEEE/ACM Transactions on*, 22(2):477–492, Feb 2014.

[2] K. Lee and M. Slaney. Automatic chord recognition from audio using a supervised hmm trained with audio-from-symbolic data. In *Proceedings of the 1st ACM Workshop on Audio and Music Computing Multimedia*, AMCMM '06, pages 11–20, New York, NY, USA, 2006. ACM.

[3] M. McVicar and T. De Bie. Enhancing chord recognition accuracy using web resources. In *Proceedings of 3rd International Workshop on Machine Learning and Music*, MML '10, pages 41–44, New York, NY, USA, 2010. ACM.

[4] M. McVicar, R. Santos-Rodriguez, Y. Ni, and T. D. Bie. Automatic chord estimation from audio: A review of the state of the art. *Audio, Speech, and Language Processing, IEEE/ACM Transactions on*, 22(2):556–575, Feb 2014.

[5] J. Morman and L. Rabiner. A system for the automatic segmentation and classification of chord sequences. In *Proceedings of the 1st ACM Workshop on Audio and Music Computing Multimedia*, AMCMM '06, pages 1–10, New York, NY, USA, 2006. ACM.

[6] Wikipedia. Hidden markov model — wikipedia, the free encyclopedia, 2014. [Online; accessed 1-December-2014].