Real-time Chord Recognition

Adam Belhouchat, Ashish Sareen

***Abstract*—Blahdy blah.**

***Index Terms*—Blah, blahdy, bloo**

I. Introduction

A. History with References

Automatic chord recognition (ACR) is the process of determining what chord is being played given an audio sample, labelling the audio sample over time with the appropriate chord [1]. A chord here is defined as two or more notes being played at the same time or close together [2]. Chord recognition has many applications, such as music segmentation or determining the similarity between two music samples [3], but one of the most attractive applications is in automatic transcription. Manual transcription can be tedious and difficult, and automatic chord recognition systems can help musicians work more quickly and more accurately [4]. Because of its wide range of applications, chord recognition has been the subject of much research in the past few decades, trying to improve the number of chords it can recognize and its accuracy through novel methods and approaches [5].

One of the first ACR systems was developed by Takuya Fujishima in 1999 [5]. He used the discrete Fourier transform (DFT) of a music sample to generate a pitch class profile, also known as chroma feature, which encodes the harmonic information of the sample over twelve different pitch classes. The chroma feature is then compared the templates, one for each of 27 different chords, and the closest match is chosen as the chord label in a process called pattern matching [6].

Since then, chord recognition has evolved greatly. Pattern matching is simple and effective, but it is not the most accurate [7]. Some researchers have found great success using hidden Markov models (HMMs) rather than template matching to perform ACR [8], and others have explored the use of deep learning and neural networks both to detect chords [9] and to extract features from the audio sample [10]. However, both HMMs and deep learning require a significant amount of training data, and good training data may be difficult to come by. Manual transcription is prone to errors due to human subjectivity [5] and as mentioned before, it can be time consuming. Nevertheless, these approaches do show significant accuracy improvements over pattern matching [7], [9]. Each of these approaches has its pros and cons, and which one is most fitting depends on the specific application.

B. Global Constraints

II. Motivation

One shortcoming of many modern chord recognition systems is the vast amount of data required to train these systems [5]. As mentioned earlier, large datasets of accurately transcribed music are not easy to find, so we wanted to develop a system that would work without training. Additionally, these data-driven approaches are opaque and it is difficult to understand how they determine the chords in an audio sample [5]. Part of our motivation for pursuing this project was to increase our skills in real-time digital signal processing and to understand how automatic chord recognition works. Throwing data at a neural network until it works does not help us understand chord recognition.

Another shortcoming of HMMs and neural networks is they are often more computationally expensive than pattern matching [1]. They work well when a music sample has been prerecorded and is later fed into the system, but they are not fast enough to perform chord recognition and transcription in real-time. We wanted to make a chord recognition system that would record musical chords and transcribe them in real-time, so we needed to approach our system differently from most modern systems.

Lastly, as mentioned earlier, we chose this project to learn more about real-time digital signal processing (DSP) and to gain more experiences with different applications of it. Chord recognition requires many DSP techniques, including sampling a signal, processing it to extract features, performing some evaluation on those features, and then returning the results of that evaluation to the user [6]. This project would give us valuable experience with all the important elements of DSP with the added constraint of performing it in real-time. This required us to perform memory management and pay close attention to our code to make sure all functions were optimized, which are valuable skills for embedded software development.

III. Approach

A. Team Organization

B. Plan and Implementation

C. Standard

The most common features used in pattern matching and HMM chord recognition systems are chroma features [11]. First introduced into chord recognition by Fujishima, basic chroma features are calculated by summing the DFT of the music signal over certain frequency bins [6]. Each chroma vector has twelve elements corresponding to the twelve pitch classes that the frequency bins are mapped to, and the distribution of energy across the twelve pitch classes determines which chord it corresponds to [7]. Because of how ubiquitous chroma features are in chord recognition, we chose to use them as our features for chord recognition.

To actually determine what chord the chroma features corresponds to, we use pattern matching, which is one of the most common methods of determining chords [11]. In

particular, we use binary pattern matching, where each chord has a corresponding template which is a specific arrangement of zeros and ones. The tones present in the ideal chord are one, while all the other tones are zero [11]. To find the matching template, many implementations look for the minimum Euclidean distance between a template and the chroma feature [1], but other approaches such as finding the minimum angle between the two are also used [7]. We chose to go with this latter approach as it gave us slightly better accuracy.

D. Theory

The key mathematical elements of our chord recognition system are the low-pass filter, the chroma features, and pattern matching. In this section, we will explain in detail the theory behind each of these steps.

The low-pass filter is generated using the bilinear transform, which converts a continuous transfer function to a discrete one. To do so, the bilinear transform uses the approximate map

$$s \to \frac{2}{T}\frac{1-z^{-1}}{1+z^{-1}} \tag{1}$$

where T is the sampling period of the discrete signal. We can then substitute this into the continuous transfer function $H_a(s)$ to get the discrete one $H_d(z)$ by

$$H_d(z) = H_a\left(\frac{2}{T}\frac{1-z^{-1}}{1+z^{-1}}\right) \tag{2}$$

We used a 6th order Butterworth low-pass filter with a cutoff frequency at 6000 Hz. This cutoff frequency preserved most of the important harmonic information while still filtering out high-frequency noise, and the order gave a reasonably steep dropoff around the cutoff frequency without being computationally expensive. The actual transfer function is too large and complex to show here, so we used a program to generate the time-domain filter function. The frequency response is given in Figure 1.

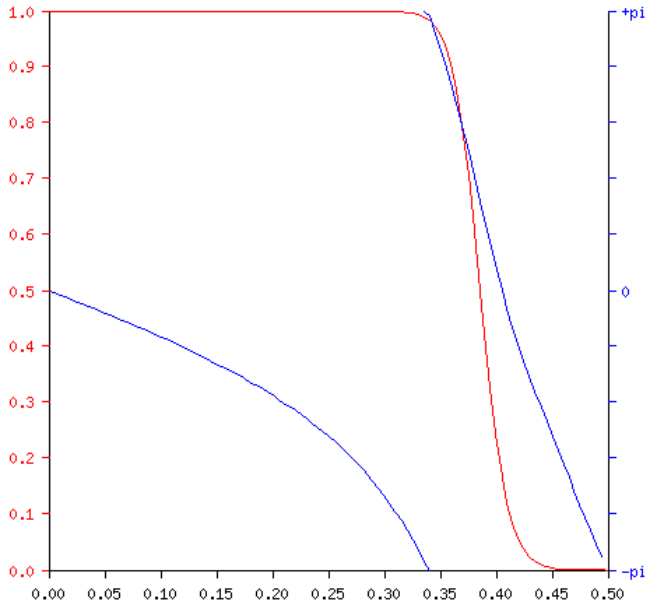

Fig. 1. Frequency response of the low-pass filter used in our chord recognition system. Courtesy of Tony Fisher at https://www-users.cs.york.ac.uk/~fisher/mkfilter/.

E. Software/Hardware

F. Operation

1) How the System was Built:

2) How to Use the System:

IV. Results

A. Description

B. Discussion

References

[1] A. M. Stark and M. D. Plumbley, "Real-time chord recognition for live performance," in *Proceedings of the 2009 International Computer Music Conference, ICMC 2009*, 2009, pp. 85–88.

[2] T. Cho and J. P. Bello, "On the relative importance of individual components of chord recognition systems," *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, vol. 22, no. 2, pp. 477–492, 2013.

[3] K. Lee, "Automatic chord recognition from audio using enhanced pitch class profile." in *ICMC*, 2006.

[4] M. Mauch, "Automatic chord transcription from audio using computational models of musical context," Ph.D. dissertation, 2010.

[5] J. Pauwels, K. O'Hanlon, E. Gómez, and M. B. Sandler, "20 years of automatic chord recognition from audio," in *ISMIR*, 2019.

[6] T. Fujishima, "Real-time chord recognition of musical sound: A system using common lisp music," *Proc. ICMC, Oct. 1999*, pp. 464–467, 1999.

[7] N. Jiang, P. Grosche, V. Konz, and M. Müller, "Analyzing chroma feature types for automated chord recognition," in *Audio Engineering Society Conference: 42nd International Conference: Semantic Audio*. Audio Engineering Society, 2011.

[8] A. Sheh and D. P. Ellis, "Chord segmentation and recognition using EM-trained hidden Markov models," 2003.

[9] N. Boulanger-Lewandowski, Y. Bengio, and P. Vincent, "Audio chord recognition with recurrent neural networks," in *ISMIR*, 2013.

[10] F. Korzeniowski and G. Widmer, "Feature learning for chord recognition: The deep chroma extractor," 2016.

[11] T. Cho, R. J. Weiss, and J. P. Bello, "Exploring common variations in state of the art chord recognition systems," 2010.