Development of an Algorithm for Real-Time Identification of Musical Chords from MIDI Signals

Introduction

Chords are collections of two or more musical notes, often played together, and are arranged in such a way that they follow the so-called "rules of harmony" (Leino, Brattico, Tervaniemi, & Vurst, 2007). These rules are recognized by humans as a response in the brain that is triggered when they are violated (Leino *et al.*, 2007). Because of this, chords are fundamental to the harmonic integrity of any musical work.

A musical chord is commonly identified by three parameters: root note, chord type, and inversion. The root note serves as a reference point for the other notes which are played in the chord. These other notes are determined by the chord type. The inversion tells which note in the chord is played lowest. For example, a C major chord, 0th inversion has "C" as the root note, and "major" as the chord type. A major chord type includes the 1st (root), major 3rd, and 5th harmonics of the typical Western major scale, and the 0th inversion says that the root note is the bottom note. Thus, the notes of the 0th inversion of a C major chord are C (1st), E (major 3rd), and G (5th), in that order.

Statement of the Problem

Humphrey, Bello, and Cho (n.d.) state that "the general music learning public places a high demand on chord-based representations of popular music" (par. 1). However, complete and accurate determination of these chords by hearing requires the use of both absolute and relative pitch, because chords utilize both an absolute reference point (root note) and a relative configuration of harmonies (chord type).

Absolute pitch is expressed when one can identify a musical note by hearing it, while relative pitch is shown when one can recognize the distances between musical notes (Zatorre, Perry, Beckett, Westbury, & Evans, 1998). While "most trained musicians" (Zatorre *et al.*, 1998) exhibit a mastery of the latter, few of them have absolute pitch.

Absolute pitch is expressed in a low percentage of the human population and acquired through a combination of favorable genes and music training at a young age (Baharloo, Service, Risch, Gitschier, & Freimer, 2000). Therefore, complete chord identification is a rare skill found in those with mastery of both absolute and relative pitch, even though chords play an important role in any musical work. An algorithm that automatically identifies chords from individual notes would be a first step towards addressing this problem.

Objectives of the Study

This study aims to develop a low-latency algorithm that correctly identifies one-root musical chords formed by playing more than two notes on a MIDI input device. Specifically, the program must identify simple & extended chords in real-time and respond quickly enough to be used in live performance (Stark & Plumbley, 2009). The program must be implemented in programming languages that have MIDI input-output libraries such as *pyrtmidi* (Kidd, 2017) for Python and *rtmidi* (thestk, 2017) for C++ to facilitate ease of coding.

Significance of the Study

Such an algorithm would be central to the development of a standalone program that comprehensively identifies chords from MIDI signal inputs, of which there are none currently available. While chord identification algorithms exist, they operate on an audio input (Fujishima, 1999; Stark & Plumbley, 2009). For those algorithms to work optimally, the appropriate audio equipment, which may not be accessible, is needed. On the other hand, many common electronic keyboards have MIDI functionality (Brown, 2016), and can thus output MIDI note signals to an outboard device such as a computer that can then run the MIDI-based chord recognition algorithm.

It would be useful in the field of music education, where a low proportion of music students have absolute pitch (Gregersen, Kowalsky, Kohn, & Marvin, 1999) despite their demand for chordal representations of music (Humphrey, Bello, & Cho, n.d., par. 1). It would allow said students to learn to identify chords more quickly and accurately, and help them develop their senses of relative and absolute pitch. It is also useful in situations when chords being played need to be verified for correctness, such as when one is learning or composing a musical piece.

Scope and Limitation

The study aims to create an algorithm that will quickly identify chords of only these types (Stark & Plumbley, 2009):

1. Basic triads (major, minor, augmented, diminished, sus2, sus4)
2. 7ths (dominant 7, major 7, minor 7, augmented 7, diminished 7, half diminished, suspended variants)
3. 9ths (dominant 9, major 9, minor 9, augmented 9, diminished 9, suspended variants)
4. 11ths (dominant 11, major 11, minor 11, augmented 11, diminished 11, suspended variants)
5. 13ths (dominant 13, major 13, minor 13, augmented 13, diminished 13, suspended variants)

Research Design

Experimental Units and details to ensure local control

Since the study involves the development of an algorithm, each experimental unit would be defined as one runtime instance of the algorithm. The evaluated runtimes will then be sorted during data collection by *input* chord type.

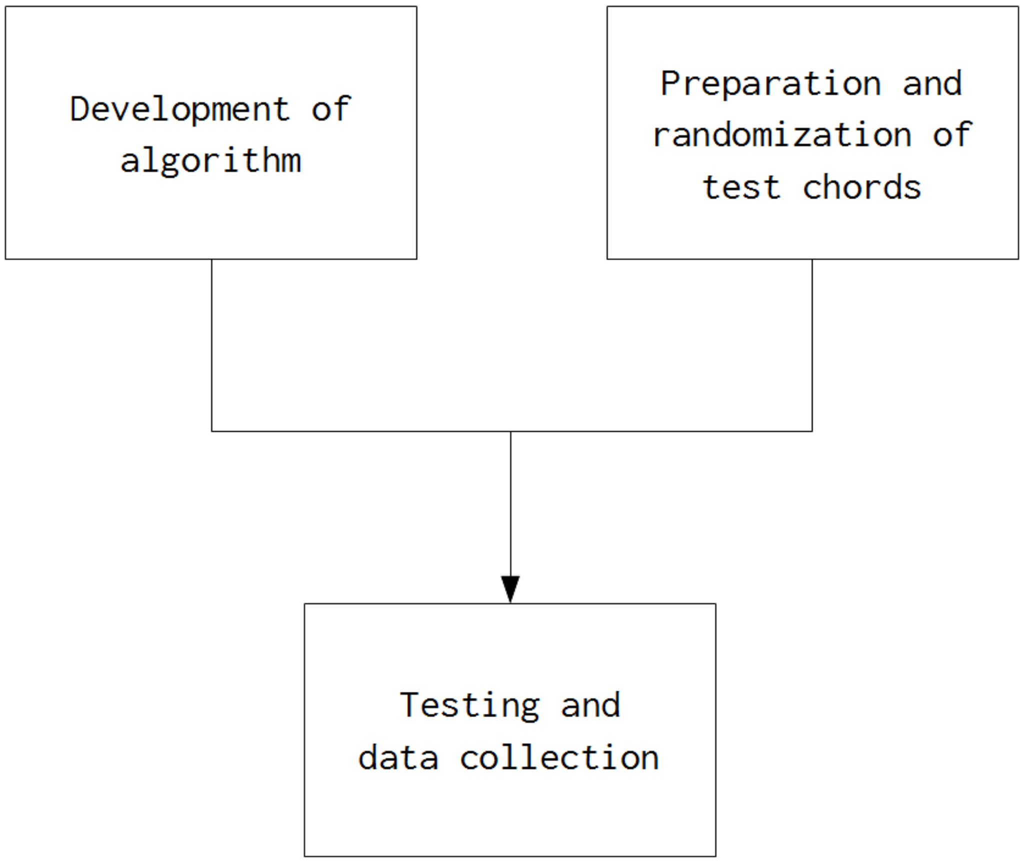
Treatments

Each treatment is a specific chord, with a root note and inversion number. A fixed number of chords will be randomly selected for every chord type by randomizing the said parameters. The chord types to be used are those found in the Scope and Limitation of this paper.

Allocation of treatments to EUs

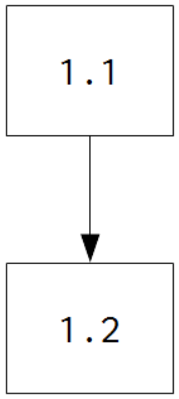
Each chord to test the algorithm through will be randomized by randomly obtaining two of the three chord parameters. These are the following: root note (C, C#, D, D#, E, F, F#, G, G#, A, A#, B), and inversion (0^{th} to $(N - 1)^{th}$), where N is the number of notes in the chord.

Level 0 diagram


Development of algorithm
Preparation and randomization of test chords
Testing and data collection

Level 1 diagrams

1 DEVELOPMENT OF ALGORITHM



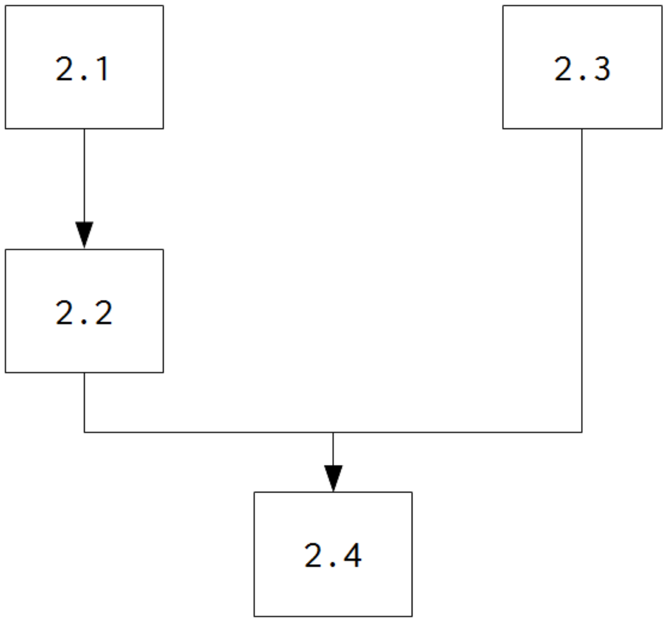
1.1 Solution process (Fujishima, 1999; Stark & Plumbley, 2009)

This involves the creation of the logic of the algorithm that will identify chords from MIDI input signals.

1.2 Implementation of algorithm in a programming language

This includes the adaptation of the logic into a runnable programming language, and the use of MIDI libraries (Kidd, 2017; thestk, 2017) *to facilitate MIDI input to the program*.

2 PREPARATION & RANDOMIZATION OF TEST CHORDS



2.1 Assignment of test chord parameters (Stark & Plumbley, 2009) to arbitrary numbers

Both root note and inversion number will be assigned arbitrary numbers to facilitate selection by the RNG in step ***2.4****.*

2.2 Determination of sample size

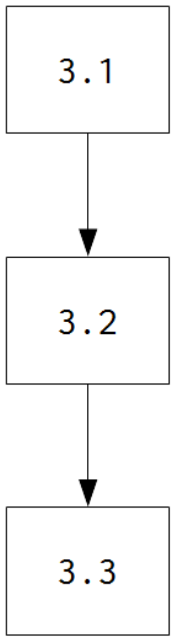
The appropriate sample size will be used to minimize bias and testing duration.

2.3 Acquisition of random number generator

2.4 Stratified random sampling of chords

For every chord type, a root note and inversion number will be randomly selected using the random number generator acquired in step ***2.3****. Each chord will be run through the algorithm and the data will be recorded in step* ***3.2****.*

3 TESTING AND DATA COLLECTION



3.1 Algorithm testing on randomly chosen chords

The algorithm proper is yet to be determined; therefore, no further elaboration can be made on this step.

3.2 Data logging

The chord notation outputs by the program will be collected alongside their respective input chords for analysis in step ***3.3****.*

3.3 Data analysis

The outputs and inputs in step ***3.2*** *will be compared and the appropriate statistical analysis will be applied.*

References

Baharloo, S., Service, S., Risch, N., Gitschier, J. & Freimer, N. (2000). Familial aggregation of absolute pitch. *American Journal of Human Genetics, 67,* 755-758. doi:10.1086/303057.

Brown, R. (2016, July 12). *Rob Brown's answer to Do all electric pianos/keyboards have MIDI?* Retrieved August 24, 2017, from https://www.quora.com/Do-all-electric-pianos-keyboards-have-MIDI.

Fujishima, T. (1999). *Realtime chord recognition of musical sound: A system using common Lisp music*. Retrieved from http://www.music.mcgill.ca/~jason/mumt621/papers5/fujishima_1999.pdf.

Gregersen, P., Kowalsky, E., Kohn, N., & Marvin, E. (1999). Absolute pitch: prevalence, ethnic variation, and estimation of the genetic component [Letter to the Editor]. *American Journal of Human Genetics, 65,* 911-913. doi:10.1086/302541.

Humphrey, E., Bello, J. P., & Cho, T. (n.d.). Chord Recognition. Retrieved from http://steinhardt.nyu.edu/marl/research/chord_recognition.

Kidd, P. (2017). *pyrtmidi: Real-time MIDI I/O for Python* [GitHub repository]. Retrieved August 23, 2017, from https://github.com/patrickkidd/pyrtmidi.

Leino, S., Brattico, E., Tervaniemi, M., & Vurst, P. (2007). Representation of harmony rules in the human brain: Further evidence from event-related potentials. *Brain Research, 1142,* 169-177. doi:10.1016/j.brainres.2007.01.049.

Stark, A. M., & Plumbley, M. D. (2009). *Real-time chord recognition for live performance* [PDF file]. Retrieved at https://www.eecs.qmul.ac.uk/~markp/2009/StarkPlumbley09-icmc.pdf.

thestk (2017). *rtmidi: A set of C++ classes that provide a common API for realtime MIDI input/output across Linux (ALSA & JACK), Macintosh OS X (CoreMIDI) and Windows (Multimedia)* [GitHub repository]. Retrieved August 23, 2017, from https://github.com/thestk/rtmidi.

Zatorre, R. J., Perry, D. W., Beckett, C. A., Westbury, C. F., & Evans, A. C. (1998). Functional anatomy of musical processing in listeners with absolute pitch and relative pitch. *Proceedings of the National Academy of Sciences, 95*, 3172-3177. Retrieved at http://www.pnas.org/content/95/6/3172.full.