RAMEAU: A SYSTEM FOR AUTOMATIC HARMONIC ANALYSIS

Pedro Kröger, Alexandre Passos, Marcos Sampaio, Givaldo de Cidra
Genos—Computer Music Research Group
School of Music
Federal University of Bahia, Brazil
{pedro.kroger,alexandre.tp,mdsmus,givaldodecidra}@gmail.com

ABSTRACT

Automated harmonic analysis is an important and interesting music research topic. Although many researchers have studied solutions to this problem, there is no comprehensive and systematic comparison of the many techniques proposed.

In this paper we present Rameau, a framework for automatic harmonic analysis we are developing. With Rameau we are able to reimplement and analyze previous techniques, and develop new ones as well. We present a performance evaluation of ten algorithms on a corpus of 140 Bach chorales. We also evaluate four of them using precision and recall and discuss possible improvements.

We also present a numeric codification for tonal music with interesting properties, such as easy transposition, preservation of enharmonic information and easy conversion to standard pitch-class notation.

1. INTRODUCTION

In music, harmonic analysis is the study of vertical sonorities and their connections, and is paramount to the understanding of tonal compositions. The analyst may find the chord roots, label sonorities with proper chords names (such as "A minor"), and identify their relationships using roman numerals.

Harmonic analysis by computer is an important, challenging, and interesting music research topic. It is challenging because the musical material has a large variety of information such as timbre, notes, rhythm, dynamics, harmony, and because music unlike image is not a linear process [15]. The harmonic changes in a choral texture, where usually all notes in all voices begin and finish at the same time, are obvious. However, chords are occasionally arppeggiated, incomplete, and intertwined with non-harmonic tones. These things contribute to increase considerably the complexity of analysis [20].

There are many practical applications for automatic music analysis, such as arranging, detection of possible logical mistakes in scores, database search, automatic accompaniment generation, and statistical analysis of musical styles for automated composition [21, 26]. Computer-based harmonic analysis is also important because it can bring new insights in music theory, in the same way the use of computer in vision and problem solving has brought new insights in these areas [26].

Although many harmonic analysis algorithms have been proposed, no single framework for comparing them has been developed. Every comparison we found in literature ([20, 2, 27, 24, 10]) is based solely on published examples, which are not a statistically representative sample. Pardo and Birmingham [20] state that "no researchers have published statistical performance results on a system designed to analyze tonal music". In this paper we will present Rameau, a framework for automatic harmonic analysis of Western tonal music developed as an attempt to solve this problem.

We have implemented ten algorithms for harmonic analysis in the framework and will present in this work a preliminary evaluation of them using a corpus of 140 Bach chorales from the 371 in the Riemenschneider edition [1]. Rameau automatically compares the answers returned by the algorithms with answer sheets prepared by musicians. Since we have answer sheets for only 140 chorales, that will be the music corpus used in this paper, although rameau can analyze all 371. While some algorithms are heavily based on previous research, other approaches, like our usage of decision trees, are new. Another significant difference between our approach and the others found in the literature is the evaluation of performance using precision and recall.

2. THE PROBLEM OF AUTOMATED HARMONIC ANALYSIS

The problem of automated harmonic analysis may better be understood if we divide it into four sub-problems.

The first problem is pitch spelling. If sharps and flats are not distinguished the program may have problems whenever enharmonic information is relevant. For example, in [21] a German augmented sixth chord is mistakenly identified as a dominant seventh chord. Most systems, such as the one found in [26], either ignore this problem or develop a pitch speller to correct the output. Most harmonic analysis software, with the notable exception of MusEs [19] and the system described in [10], use input formats that lose enharmonic information.

The second problem, the segmentation problem [20], is to determine what are the vertical sonorities in a piece

of music and group these sonorities in harmonically significant segments, or chord-spans. The main reason for the complexity of segmentation is that the number of possible segmentations of a piece is roughly an exponential function of its length, making a thorough evaluation of all possibilities impractical [21].

The third problem is labeling segments with chord names. Not every segment forms a chord, some will consist solely of non-chord tones and other melodic information. Labeling might require contextual information, which complicates the matter a bit, making local decisions difficult. Still, over 80% of accuracy is possible ignoring context (see section 5).

The last problem is finding the main key and the modulations of the piece and assigning a tonal function to each chord. This has been explored by [22, 13, 27, 28] and others, with varying degrees of success. Currently, as our research is still in its infancy, we do not approach this problem.

3. NUMERIC CODIFICATION FOR TONAL MUSIC

The pitch-class notation, or some variation such as MIDI, is probably the most used way to numerically represent pitch. The main problem with this notation is the loss of enharmonic spelling. In this section we will revise three notations created independently by Brinkman, Hewlett, and Oliveira [4, 9, 18] to address this problem. For the sake of simplicity we will refer to them as binomial, base-40, and base-96, respectively.

In the binomial notation notes are represented as tuples in the form of <pc,nc>, where pc stands for pitch-class (0 to 11) and nc for note-class (0 to 7). Note-class 0 means the note is written as accidentals over C, 1 represents accidentals over D, and so on. For example, C♯ is represented as <1,0>, D♭ as <1,1>, D♯ as <3,1>, and C♭ as <11,0>. In this last example 11 represents the pitch-class of the note and 0 indicates how it will be spelled. The binomial notation also allows a packaged representation using only single numbers, where the pitch-class is multiplied by 10 and added to the note-class. For example, C♯ becomes 10 and D♭ 11.

Unlike the binomial notation, the base-40 notation uses only single numbers. The octave is divided linearly in 40 notes from C𝄫 to B𝄪. Table 1 shows a few notes in this system.

The base-96 notation is simple, elegant, and overcomes a few shortcomings in both the binomial and base-40 notations. As the original publication of the base-96 notation is available only in Portuguese, we will describe it here, comparing it with the other two.

The base-96 is also a single number notation. It has all the qualities of the base-40 notation with some additional advantages. Table 2 shows how notes are encoded. In this notation the intervals are invariant under most transformations, such as inversion and transposition. Table 3 shows the coding for the intervals. The first column indicates the interval name; P, M, m, d, A for perfect, major, minor, diminished, and augmented, respectively. The letters before d and A indicate the quantity of the interval. For instance, tA is triple-augmented.

note	code
c𝄫	1
c♭	2
c	3
c♯	4
c𝄪	5
–	6
d𝄫	7
...	...

Table 1. A few notes in the base-40 notation

	c	d	e	f	g	a	b
7♭		7	21			62	76
6♭	90	8	22	35	49	63	77
5♭	91	9	23	36	50	64	78
4♭	92	10	24	37	51	65	79
3♭	93	11	25	38	52	66	80
2♭	94	12	26	39	53	67	81
♭	95	13	27	40	54	68	82
♮	0	14	28	41	55	69	83
♯	1	15	29	42	56	70	84
2♯	2	16	30	43	57	71	85
3♯	3	17	31	44	58	72	86
4♯	4	18	32	45	59	73	87
5♯	5	19	33	46	60	74	88
6♯	6	20	34	47	61	75	89
7♯				48			

Table 2. Notes in the base-96 notation.

The base-96 notation works for up to seven flats and sharps, which is more than enough for tonal music. This notation is compatible with the pitch-class system (modulo 12). In the base-96 notation G = 55, and the result of 55 mod 12 is 7, representing G in the pitch-class notation.

The problem with the base-40 notation is that it does not work well when there are more than two accidentals. For example, the interval between C𝄫 and C𝄪 (a quadruple augmented unison) has the same value (6) as diminished second. On the other hand, in the base-96 notation this interval can be correctly computed. In our corpus this would not have direct consequences, but many algorithms, such as Pardo et al's, rely on transposing the melodies, and this could create some issues.

The main problem with the binomial notation is that operations that are (and should be) simple become complex. In both base-40 and base-96 notations, transposition is as simple as adding an index to a note, while in the binomial system it requires two different operations (mod and div) besides the addition. Its packaged representation is supposed to simplify things, but as an example, transposition becomes something like:

	1^{st}	2^{nd}	3^{rd}	4^{th}	5^{th}	6^{th}	7^{th}	8^{th}
sd		7	21	35	49	62	76	90
qd		8	22	36	50	63	77	91
qd		9	23	37	51	64	77	92
td		10	24	38	52	65	78	93
dd		11	25	39	53	66	79	94
d		12	26	40	54	67	80	95
m		13	27			68	82	
P	0			41	55			96
M		14	28			69	83	
A	1	15	29	42	56	70	84	
dA	2	16	30	43	57	71	85	
tA	3	17	31	44	58	72	86	
qA	4	18	32	45	59	73	87	
qA	5	19	33	46	60	74	88	
sA	6	20	34	47	61	75	89	
oA				48				

Table 3. Intervals in base-96 notation

$$10 \times ((a \div 10 + B \div 10) \mod 12) + ((a \mod 10 + B \div 10) \mod 7) \quad (1)$$

To do anything non-trivial, like interval-class vectors, yet extra work has to be done, increasing complexity.

We believe the base-96 codification is a good form of describing tonal music numerically and it should be more known among researches and developers. For these reasons we have adopted it, although no algorithm implemented in rameau makes direct use of it as of the writing of this paper [1] , since they are reimplementations of algorithms that expect pitch classes as input.

4. THE RAMEAU FRAMEWORK

To properly study, understand and compare algorithms for automated harmonic analysis we have designed and implemented a framework, Rameau, that should enable us to

1. compare results precisely and reproducibly with manual analysis of a large corpus of music;
2. allow algorithms to access detailed information in the input score, such as enharmonic differences and meter; and
3. easily develop new algorithms, test existing ones, and precisely compare the results.

Reimplementing existing algorithms in the literature allows us to easily evaluate their accuracy, study their errors and compare their merits and flaws. We are unaware of any systematic comparison of different approaches to harmonic analysis, similarly to what Gomez and Herrera [7] do for tonality recognition.

Rameau is written in Common Lisp and runs on the Steel Bank Common Lisp [5] and CMUCL [12] compilers and the Clisp [8] interpreter on the GNU/Linux and Microsoft Windows operating systems. Increased portability is one of our goals.

4.1. Architecture

The architecture of the Rameau framework is simple. First, a score is parsed into a list of notes, which is then split into a list of sonorities. Then, these sonorities are sent to each algorithm for analysis. The analysis results and their comparison with the answer sheets are then output either textually or as an annotated score, as in figures 1 and 4.

The algorithms used for analysis are implemented using a very simple Common Lisp API. To implement an algorithm it is only necessary to place a lisp file in a special directory and call the `register-algorithm` function inside that file, specifying the algorithm's name, an analysis function and a comparison function as parameters. The input to the analysis function is a list of sonorities. In Rameau, sonorities are lists of notes, each note specifying its onset time, duration, octave and pitch. The output is a list of either chords or non-chord tones, one for each input sonority. The comparison function is responsible for determining which parts of the chord are to be considered when assessing correctness of the analysis. This is necessary because, for example, Temperley and Sleator's [26] algorithm does not identify a chord's mode. Chords are represented internally as structures that may have a root, a mode, a seventh, a bass, and other additions. The framework also provides a rich library to deal with common musical operations.

Using this API we have implemented:

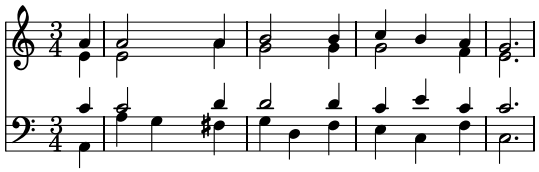
1. a subset of the algorithm described in [21] (ignoring, for now, segmentation),
2. a work-in-progress port of the algorithm described in [26],
3. four neural networks (using the Fann [17] library) roughly similar to some described in [27], and
4. four decision trees modeled after the neural networks (using code from [14]).

Rameau's source code is publicly available, under a GNU GPL [6] license, in a git [3] repository at `git://genos.mus.br/rameau.git` and for visualization at `http://git.genos.mus.br/?p=rameau.git`.

4.2. The interface

Rameau is a command line program, although a GUI version is planned. The user can select the chorales to be analyzed and specify the algorithms for the analysis. The output shows a table with the partition number, the correct

[1] Since then, every algorithm has been extended or reimplemented to incorporate this codification.



Partitions:	1	2	3	4	5	6	7	8	9	10	11
Pardo-Birm.:	Am	Am	***C***	D	G	G	G7	C	***C***	F	C
Mode Tree:	Am	Am	***C/G***	D/F#	G	***D***	G7/F	C/E	C7+	F	C
Chord Tree:	Am	Am	***C/G***	D/F#	G	***D***	G7/F	C/E	C7+	F	C
Context Tree:	—	A	A	D	G	***D***	G	C	C	F	C
Simple Tree:	A	A	A	D	G	***D***	G	C	C	F	C
Mode Net:	Am	Am	Am7/G	D/F#	G	G/D	G7/F	C/E	C7+	F	C
Chord Net:	Am	Am	Am7/G	D/F#	G	G/D	G7/F	C/E	C7+	F	C
Context Net:	A	A	A	D	G	G	G	C	C	F	C
Simple Net:	A	A	A	D	G	G	G	C	C	F	C
Temper.-Sle.:	A	A	A	***A***	***A***	***A***	***A***	***A***	***A***	***A***	***A***
Answer:	Am	Am	Am7/G	D/F#	G	G/D	G7/F	C/E	C7+	F	C

Figure 1. An analyzed excerpt from chorale 12, "Puer natus in Bethlehem"

analysis for each sonority (taken from the answer sheet) and the analysis output for each chosen algorithm. The nice thing about this output is that it can be further processed using regular unix tools such as grep, awk, and sed.

To simplify reading the results, rameau can output a score with the chorale and the analysis from each algorithm chosen. The music notation program LilyPond [16] is used to render the score.

Figure 1 shows the result of the analysis of the first phrase of chorale #12. The first line shows the partition number, the last line, the expected answer, and the lines in between, the analysis results for the chosen algorithms. The output score shows incorrect analysis in bold italic. Non-chord tones are notated as "—". It is important to remember that while some algorithms identify chords, others identify only the root.

Rameau performs the analysis of one chorale, using all stable algorithms, in less than a half-second on a Pentium Celeron M at 1.4 GHz with 2 GB RAM linux box. All 140 chorales are analyzed and the results compared with the answer sheets under 40 seconds.

4.3. Test corpus

We are building a corpus of analyzed and digitalized Bach chorales to use as training and test data. Bach chorales were chosen because:

1. they are easily analyzed. For example, the segmentation problem in them consists simply of determining the sonorities.

2. Their chord density is high — there are many more chords per measure than in a symphony, for example.

3. They are canonical examples of tonal harmony.

4. There are 371 on the Riemenschneider edition, more than enough to train our algorithms and get precise error analysis.

5. Their texture is simple and constant. It consists basically of four voices forming simple triads and tetrads.

6. Although they have been widely used as examples [24, 27, 10, 29], no single research has analyzed all Bach chorales. We believe doing this will create a useful corpus of harmonic information.

We have answer sheets for 140 chorales and plan on fully incorporating every chorale in the Riemenschneider [1] edition soon. The corpus is stored in a subset of the GNU LilyPond format, from which we generate MIDI files and typeset scores, possibly annotated with analysis results (both our answer sheets and computer-made results). The LilyPond format was chosen because it is easy to parse, easy to read and write, and compact (unlike MusicXML). Also, there are tools to convert from different formats (such as MIDI) to it and LilyPond itself can be used to typeset the scores and export them to MIDI.

When we have answer sheets for all chorales we plan on incorporating the Kostka-Payne [11] corpus, Beethoven sonatas, Bach cello suites and other pieces.

4.4. The answer sheet format

The results of manual analysis performed on the chorales are stored in a simple and flexible text format. It is designed to be as close as possible to usual popular notation and represent inherent ambiguities in analysis and non-chord tones.

The first four sonorities of the answer sheet for Bach's chorale #1 "Aus meines Herzens Grunde", for example, are stored as `G G C/E (C7+/E [b])`. Each chord symbols represents a sonority. Chords in parenthesis represent possible interpretations of a single sonority. Notes in brackets are non-chord tones, marking sonorities that do not constitute chords.

This information is then used as answer sheets and training data on the many algorithms implemented in our system.

5. PRELIMINARY RESULTS

5.1. Pardo and Birmingham's algorithm

The algorithm described in [20] has some interesting properties. It handles most simple examples of tonal harmony really well, but has no clear notion, by design, of non-chord tones, augmented chords and other possible analysis. We have found no need, as of yet, of implementing the segmentation algorithm, since segmentation is trivial in Bach Chorales.

The current accuracy is $66 \pm 9\%$.

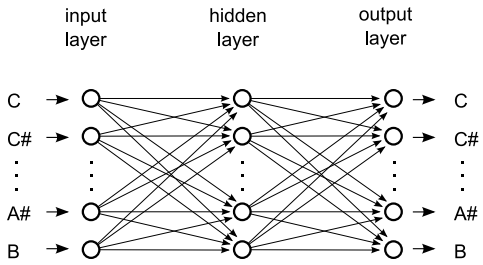


Figure 2. The Simple-net

5.2. Temperley and Sleator's algorithm

We have reimplemented the algorithm used in Temperley and Sleator's Melisma Music Analyzer and described in [26]. We also found their implementation of their algorithm, but neither their program nor our reimplementation was able to match the performance claimed in the article. Tsui [27] was also unable to reproduce their results.

The Melisma algorithm is brittle and depends on many parameters. Its performance is orders of magnitude slower than any other we have evaluated. Our reimplementation is also fragile and occasionally fails to process a chorale. Its accuracy is around $50 \pm 20\%$. The original implementation's accuracy couldn't be computed due to difficulties in parsing Melisma's output and matching it with our answer sheets. On the scores we manually matched, however, the accuracy seems poor, as can be seen in figures 1 and 4.

5.3. Neural networks

Neural networks are tools for non-linear statistical data modeling that work by having artificial neurons exchange information. There are many varieties of neural networks, each being useful for a certain type of problem. Our networks are all multilayer feed-forward neural networks, with one hidden layer each. This is the standard model for pattern recognition [23].

We have implemented in Rameau four neural-network-based algorithms for harmonic analysis, basing our approaches on [27]. The input of our simplest algorithm, `Simple-net`, is how many times each pitch class sounds in a sonority (its pitch count). As output, the network activates the neuron representing the root for that sonority (or does nothing, if the sonority does not form a chord). Figure 2 illustrates its connections and labels. To study the effect of contextual information in harmonic analysis we have also implemented `Context-net`, differing from the `Simple-net` only by also looking at the pitch counts of two preceding sonorities and one immediately the one being analyzed. `Context-net` and `Simple-net`'s performances are equivalent, so perhaps the contextual information necessary for harmonic analysis is not easily inferrable from pitch counts. The two other neural networks, `Chord-net` and `Mode-net`, also determine a chord's mode and its seventh. `Chord-net`'s input is the same as `Simple-net`'s, while `Mode-net` also sees the results of `Context-net`'s analysis for each sonority.

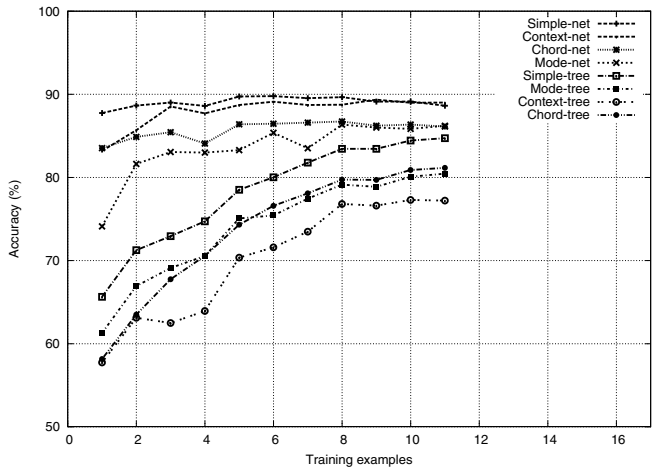


Figure 3. Accuracy versus amount of training data per algorithm

Our current accuracies for these algorithms are $88 \pm 8\%$ for `Simple-net`, $87 \pm 8\%$ for `Context-net`, $83 \pm 8\%$ for `Chord-net` and $83 \pm 8\%$ for `Mode-net`.

5.4. Decision trees

Decision trees are useful data modeling tools, with many uses in the machine learning community [14, 23]. They are most useful when trying to extract meaningful patterns from data in a human-readable way. We have built four decision trees that perform harmonic analysis, each mirroring one of the neural networks. `Simple-tree` looks at the pitches in a sonority and outputs the sonority's root. `Context-tree` looks also at the pitches in a few surrounding sonorities. `Chord-tree` looks at the pitches of a sonority and outputs its mode. `Mode-tree` looks at the pitches of a sonority (transposed to match `Simple-tree`'s analysis for that sonority) and outputs only the mode, which is matched with `Simple-tree`'s root to get a result.

Our current accuracies for the decision-tree based algorithms are $82 \pm 8\%$ for `Simple-tree`, $76 \pm 9\%$ for `Context-tree`, $77 \pm 10\%$ for `Mode-tree` and 50 ± 11 for `Chord-tree`.

5.5. The effect of training set size

The effect of training set size on the accuracy of the neural networks and decision trees can be seen in figure 3. One chorale is enough information for `Simple-net`. Most decision-tree algorithms need far more chorales than their neural-network counterparts to perform similarly, due to their lower generalization performance.

5.6. Performance analysis

Harmonic analysis can be seen as the process of retrieving chordal and functional information from a tonal piece. Information retrieval is a fertile research area, and has developed many tried and true techniques and models. For

	°	°7	ø7	m	m7	M	7	7+	+	nct	inc	avg
PB	63	59	37	77	0	66	72	0	0	0	0	34
MT	56	54	66	78	61	87	76	27	0	85	23	56
CT	56	54	66	78	67	87	76	27	0	85	23	56
MN	67	0	72	89	70	92	85	34	0	84	0	54
CN	70	0	71	87	75	90	89	35	0	86	18	56

PB: Pardo-Birmingham, MT: Mode tree, CT: Chord tree, MN: Mode net, CN: Chord net

Table 4. Precision (%)

Chord	°	°7	ø7	m	m7	M	7	7+	+	nct	inc
Freq.	2.8	0.7	2.3	16.4	4.6	37.6	9.6	1.0	0.1	24.3	0.3

Table 5. Frequency of chords in our corpus (%)

this reason we chose to analyze our results in terms of the two most used metrics in the information retrieval community: precision and recall [23].

Precision measures how correct are the returned results, whereas recall measures how likely is a sonority's mode to be correctly recognized. While a combined measure of these characteristics (like average correctness over all sonorities) is a good assessment of the general performance of an algorithm, it disregards a lot of interesting information about its behavior, so it is interesting to consider them separately.

Tables 4 and 6 show that with proper modifications to recognize more chord modes Pardo and Birmingham's algorithm can match or even exceed the accuracy of our current machine learning techniques. Table 6 shows that our machine learning algorithms are missing many melodic notes. Tables 4 and 6 also indicate that the neural network algorithms are failing to recognize fully diminished, incomplete, and augmented chords, probably because our training corpus lacks enough examples of these chords, as seen in table 5. Because the decision trees are better at these rare chords we can conclude that they recall few examples better, while the neural networks tend to generalize more efficiently when more examples are available.

As we implement more algorithms, performance analysis by rigorous metrics will enable us to draw better conclusions about the merits and flaws of each technique.

	°	°7	ø7	m	m7	M	7	7+	+	nct	inc	avg
PB	86	85	77	91	0	97	93	0	0	0	0	48
MT	68	36	44	89	65	94	82	59	0	55	18	55
CT	68	37	45	88	66	94	82	56	0	58	18	56
MN	81	0	61	93	82	94	90	70	0	68	0	58
CN	86	0	68	93	77	95	88	78	0	67	5	60

PB: Pardo-Birmingham, MT: Mode tree, CT: Chord tree, MN: Mode net, CN: Chord net

Table 6. Recall (%)

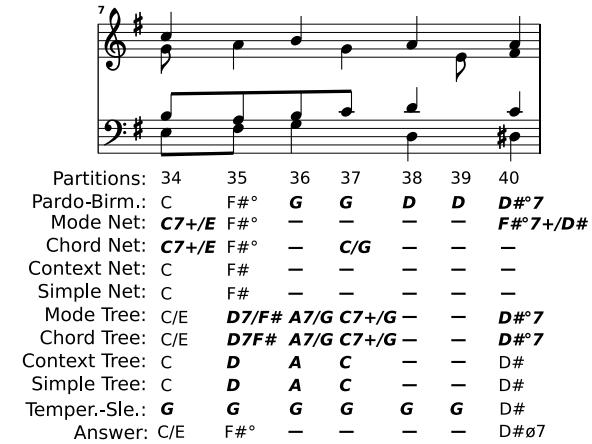


Figure 4. An analyzed excerpt from chorale 54, "Lobt Gott, ihr Christen, allzugleich"

5.7. Analysis example

The chorale excerpt in fig. 4 is a little tricky to analyze because all chords in partitions 36–39 have a non-chord tone. The A in partition 36 is a suspension, the C in p. 37 a passing note, the G in p. 38 another suspension, and the E in p. 39 a cambiata. The neural network algorithms correctly identify the non-chords tones, the decision tree algorithms fail in partitions 36 and 37 but give a correct answer in p. 38–39, and pardo-birmingham is able to identify the basic harmonic framework (without non-chord tones, however), while temperley-sleator incorrectly assumes all chords but the last have G as root.

6. CONCLUSIONS AND FUTURE WORK

In this paper we described Rameau, a framework for automatic harmonic analysis of tonal music designed to allow an easy comparison between algorithms and results. We described the problem involved in analyzing harmony with a computer and presented a numeric codification for tonal music with interesting properties and easy conversion to and from standard pitch-class notation. We then analyzed and compared the results of ten algorithms implemented in Rameau using a corpus of 140 Bach chorales.

Although some algorithms in Rameau are heavily based on previous research, our usage of decision trees is new. Some of our algorithms using decision trees have a better accuracy than previous work [20, 26]. Another significant difference on our approach is the evaluation of performance using precision and recall.

Currently (as of the publication of this paper) we have almost finished reimplementing Maxwell's algorithm, we have implemented a k-nearest-neighbor classifier and all algorithms described here have been extended to incorporate the base-96 codification.

A few important algorithms, like Ulrich's algorithm [28], Taube's MTW [24], Raphael and Stoddard's hidden Markov model [22], and many others are still unimplemented in Rameau. We plan on implementing seg-

mentation, which will allow us to extend our corpus of music and answer sheets, incorporating, for example, the Kostka-Payne corpus [25]. We are looking for new metrics and evaluation techniques, and will extend and improve our existing algorithms accordingly. We also intend to implement functional and non-chord tone analysis.

We believe all these tasks will be considerably easier now our basic platform is mostly finished.

7. ACKNOWLEDGEMENTS

This project is sponsored by FAPESB (Fundo de Amparo à Pesquisa do Estado da Bahia). We's like to thank Iara Malbouisson, Jamary Oliveira and Christina von Flach for helpful advice in preparing this text.

8. REFERENCES

[1] Johann Sebastian Bach. *371 Harmonized Chorales and 69 Chorale Melodies with figured bass*. G. Schirmer, New York, 1941.

[2] Jérôme Barthélemy and Alain Bonardi. Figured bass and tonality recognition. In Stephen Downie and David Bainbridge, editors, *Proceedings of ISMIR 2001*, pages 129–136, Bloomington, Indiana USA, October 2001.

[3] Petr Baudis. Git user's manual. Available at `www.kernel.org/pub/software/scm/git/docs/user-manual.html`, s.d.

[4] Alexander R. Brinkman. A binomial representation of pitch for computer processing of musical data. *Music Theory Spectrum*, 8:44–57, 1986.

[5] SBCL development team. Sbcl 1.0.13.45 user manual. Available at `www.sbcl.org/manual/`, 2007.

[6] Free Software Foundation. GNU general public license. Avalilable at `www.gnu.org/licenses/gpl-3.0.txt`, Jun 2007.

[7] Emilia Gomez and Perfecto Herrera. Estimating the tonality of polyphonic audio files: Cognitive versus machine learning modelling strategies. In Claudia Lomeli Buyoli and Ramon Loureiro, editors, *Proceedings of ISMIR 2004*, pages 92–95, Barcelona, Spain, October 2004. Universitat Pompeu Fabra.

[8] Bruno Haible, Michael Stoll, and Sam Steingold. Clisp. Available at `clisp.cons.org`, 2007.

[9] Walter B. Hewlett. A base-40 number-line representation of musical pitch notation. *Musikometrika*, 4:1–14, 1992.

[10] Plácido R. Illescas, David Rizo, and José M. Iñesta. Harmonic, melodic, and functional automatic analysis. In *Proceedings of the 2007 International Computer Music Conferrence*, pages 165–168, 2007.

[11] Stefan M. Kostka, Dorothy Payne, and Allan Schindler. *Tonal harmony, with an introduction to twentieth-century music*. McGraw-Hill, Boston, 5th edition, 2003.

[12] Robert A. MacLachlan. Cmucl user's manual. Available at `common-lisp.net/project/cmucl/doc/cmu-user/`, Nov 2006.

[13] H. John Maxwell. An expert system for harmonizing analysis of tonal music. In K. Ebcioglu, O. Laske, and M. Balaban, editors, *Understanding Music with AI: Perspectives on Music Cognition*, pages 335–353. AAAI Press, 1992.

[14] Tom M. Mitchell. *Machine Learning*. McGraw-Hill, New York, 1997.

[15] Rémy Mouton and François Pachet. Numeric vs symbolic controversy in automatic analysis of tonal music. In *IJCAI'95 Workshop on Artificial Intelligence and Music*, pages 32–40, 1995.

[16] Han-Wen Nienhuys and Jan Nieuwenhuizen. Lilypond. Available at `www.lilypond.org`, Jan 2008.

[17] Steffen Nissen. Fast artificial neural network library. Available at `leenissen.dk/fann/html/files/fann-h.html`.

[18] Jamary Oliveira. Em busca de uma codificação. *Cuadernos Interamericanos De Investigación en Educación Musical*, 1(2), 2001.

[19] François Pachet, Geber Ramalho, Jean Carrive, and Guillaume Comic. Representing temporal musical objects and reasoning in the MusES system. *Journal of New Music Research*, 5(3):252–275, 1996.

[20] Bryan Pardo and William P. Birmingham. Automated partitioning of tonal music. Technical report, Electrical Engineering and Computer Science Department, University of Michigan, 1999.

[21] Bryan Pardo and William P. Birmingham. Algorithms for chordal analysis. *Computer Music Journal*, 26(2):27–49, 2002.

[22] Christopher Raphael and Josh Stoddard. Harmonic analysis with probabilistic graphical models. In Holger H. Hoos and David Bainbridge, editors, *Proceedings of ISMIR 2003*, Baltimore, Maryland, USA, October 2003. The Johns Hopkins University.

[23] Stuart Russell and Peter Norvig. *Artificial Intelligence: A Modern Approach*. Prentice Hall, 2nd edition, 2002.

[24] Heinrich Taube. Automatic tonal analysis: Toward the implementation of a music theory workbench. *Computer Music Journal*, 23(4):18–32, 1999.

[25] David Temperley. Bayesian models of musical structure and cognition. *Musicae Scientiae*, 8(2):175–205, 2004.

[26] David Temperley and Daniel Sleator. Modeling meter and harmony: a preference-rule approach. *Computer Music Journal*, 23(1):10–27, 1999.

[27] Wan Shun Vincent Tsui. Harmonic analysis using neural networks. Master's thesis, University of Toronto, 2002.

[28] John Wade Ulrich. The analysis and synthesis of jazz by computer. In *Proc. of the 5th IJCAI*, pages 865–872, Cambridge, MA, 1977.

[29] Terry Winograd. Linguistics and the computer analysis of tonal harmony. *The Journal of Music Theory*, 12:2–49, 1968.