AUTOMATIC CLASSIFICATION OF GUITAR PLAYING MODES

Raphael Foulon, Pierre Roy, François Pachet[1],

[1] Sony Computer Science Labs
foulon@csl.sony.fr

Abstract. Musicians typically use various playing modes (bass lines, chord comping, solo melody improvisation) when they perform with their instrument. The design of augmented instruments calls for a precise understanding of such playing modes, from real-time analysis of musical input. In this context, the accuracy of mode classification is critical because it underlies the design of the whole interaction taking place. In this paper, we present an accurate and robust mode classifier for guitar audio signals. Our classifier distinguishes between 3 modes routinely used in jazz improvisation: bass, solo melodic improvisation and chordal playing. Our method uses a supervised classification technique applied to a large corpus of training data, recorded with different guitars (electric, jazz, nylon strings, electro-acoustic). We detail our method and experimental results over various data sets. We show in particular that the performance of our classifier is comparable to that of a MIDI-based classifier. We discuss the application of the classifier to live interactive musical systems. We discuss the limitations and possible extensions of this approach.

Keywords: Audio classification, playing mode, guitar.

1 Introduction

Interactive musical applications require some form of control from the player. Control can be expressed explicitly by the musician with controllers (faders, knobs or pedals). However, many control information cannot be given explicitly, because it creates a cognitive overhead that interferes with the performance. Implicit control information can be given through the automatic analysis of the audio signal of an instrument. Such information provides the system with valuables clues about the intention of the musician. For instance, in jazz improvisation, a musician typically alternates between several *playing modes* such as chord comping, solo melodic improvisation, walking bass, etc. Recognizing these modes automatically opens the way to musically aware interactive applications. In this paper, we address the problem of automatically classifying the audio input of a guitar improviser into such musical modes. The objective of playing mode classification is to build robust interactive musical applications (for instance, performance oriented ones, such as the VirtualBand [10], augmented instruments, etc.). We discuss some of these applications with more details later in this article.

Playing mode analysis has so far been studied by the MIR community essentially from the viewpoint of *expression modeling*. As a consequence, most of the works in

playing mode identification focus on timbral aspects of a given playing technique. For instance, [8] and [11] propose a system which classifies in real time different *playing techniques* used by a guitarist. These techniques include "up and down legatos", "slides", "slapped / muted notes", as well as the position of the pick on the neck with regards to the bridge. The system relies on the analysis of both the incoming audio signal and/or gesture capture data. Similar topics have been investigated with the goal of modeling *expressivity*, such as the articulation in nylon guitar [9]. [1] presents a feature-based approach to classify several plucking styles of bass guitar isolated notes, and [2] describes an automatic classifier of guitar strings for isolated notes using a feature-based approach, and a two-step analysis process. [19] studies the retrieval of played notes and finger positions from guitar audio signals. Instrumental techniques classification methods have been investigated for the beatbox [13] and [14] and the snare drums [16] with some success. [15] describes an approach to analyze automatically audio *effects* applied to an electric guitar or bass, and [18] studied the automatic classification of guitar tones. As we will see below, their objective is in some sense opposite to ours, since we aim at extracting information from the guitar signal that is precisely timbre-independent.

Our objective is different. For instance, in guitar, jazz in particular, there are many ways to play: octave playing (typical of Georges Benson or Wes Montgomery styles), chord comping, bass lines, mix of chords and bass (as in Bossa nova), etc. We call these ways of playing the guitar playing modes. Our aim is to precisely to detect these playing modes in real time for interactive applications. In other words, we don't aim at extracting information about the way the musician plays musical phrases (using staccato, legato, slap…), we want to extract information about the *musical content* that is played (notes, chords…). Playing modes would be in principle easy to analyze from a score of the performance. However, score-related symbolic data (pitches, duration of the notes…) are available only from MIDI instruments. Furthermore, the accuracy of MIDI guitar is not perfect, and requires specific, expensive hardware. One way to extract our information would be to use automatic score transcription from audio [6] [12]. However, these techniques are not accurate enough to build robust live systems. More specifically, [5] addresses the problem of guitar audio transcription, but this approach does not take into account the dynamic variation that can occur in live recordings, and is biased by the use of a unique guitar model for training and testing.

One key problem is to detect accurately and robustly polyphony from the guitar signal. Monophony vs. polyphony classification has been investigated by [7], using the YIN pitch estimation algorithm [3] with bivariate Weibull models. This method is practical since it only requires short training sets (about 2 minutes of audio), works for many instruments, and achieves good performance (a 6.3% global error rate). Most importantly, this work shows that the YIN descriptor yields informative features for polyphony detection.

In this paper, we describe a mode classifier that works directly from a guitar audio signal, and classifies it into three basic playing modes described above: bass, chords and melody. Following [2] the classifier is based on a 2-step analysis process (single frames then smoothing on larger windows), based on YIN-derived features, pitch and inharmonicity indicator like [7] and [19].

Our classifier is largely timbre-independent, i.e. works well with several types of guitar. We describe the training data in the next section. The classifier is described in

Section 3 and its performance discussed in Section 4. Section 5 describes applications of the classifier for interactive music systems.

2 Datasets

All guitars exploit the sound coming from the vibration of strings. But there are many types of guitars and therefore many different guitar sounds. In order to avoid bias or over fitting due to the use of a single guitar for training data, we built an audio dataset recorded with 4 different guitar types: a Godin LGX-SA solid-body guitar (*God*) – which has also a MIDI output – which output has been fed to a AER Compact 30 jazz amplifier, a Cort LCS-1 jazz guitar (*Cort*), an Ovation Electric Legend (model Al Di Meola) electro-acoustic guitar (*Ovat*) and a Godin Nylon SA nylon string guitar (*Nyl*) (see Fig.1).

Fig.1. The 4 guitar types used for our datasets. *Top-left*: one is a pure solid-body (Godin LGX), two of them have a hollow-body (the Cort jazz archtop and the Ovation electro-acoustic), and one of them has nylon strings (Godin Classic).

Each subset contains the recordings of seven jazz standards: *Bluesette*, *The Days of Wine and Roses*, *LadyBird*, *Nardis*, *Ornithology*, *Solar*, and *Tune Up*. Each song has been recorded three times, for each playing mode: *melody*, *bass*, and *chords*. The database contains therefore 84 files (1 hour and 39 minutes of audio) which are all available at *http://flow-machines.com/mode_classification_sets*.

3 The mode classifier

Our method uses a two-phase analysis: first, short signal frames (50ms) are classified with a supervised classification algorithm, which determines playing mode over short time windows, with an imperfect accuracy. Then, information obtained over the 50ms frames is aggregated to classify a whole audio chunk. The scheme of the algorithm is shown in Figure 3.

3.1 Feature selection

We use a training set containing jazz guitar recordings, which have been played and tagged in each of our three playing modes: "bass", "melody" and "chords". We describe the data sets in more details in Section 4.

As a first step, we performed feature selection to determine which features are the most relevant for our problem. We used the Information Gain algorithm [17] of Weka [20], set with the lowest threshold possible (-1.8×10^{308}) to obtain a list of features ranked by information gain, and ran it on a set of 37 features divided in two sets:

1) Basic audio features: MFCC (13), harmonic-to-noise ratio, spectral centroid, spectral flatness, spectral kurtosis, spectral decrease, spectral spread, spectral rolloff, spectral skewness, chroma (12), RMS.

2) YIN features, following [5]: YIN pitch, YIN inharmonicity indicator and YIN variance.

Feature selection yields the 6 following features: harmonic-to-noise ratio, YIN pitch and YIN inharmonicity, spectral spread, spectral centroid and spectral kurtosis. This confirms that YIN features are indeed interesting for our task. To further reduce the feature set, we retained only the 4 following features:

1) YIN pitch, which was quantized to avoid overfitting (this point is explained below), computed with an absolute threshold of 0.2 for aperiodic/total ratio,

2) YIN inharmonicity coefficient, computed in the same manner,

3) Harmonic-to-noise ratio of the signal, computed with a fundamental frequency of 185 Hz (which is the lowest frequency possible with 50ms frames, we would have to work with larger frame lengths to decrease the fundamental),

4) Spectral spread.

3.2 Frame selection and training

The training sets are normalized, and sliced into 50ms frames, overlapping at 75%. We chose this duration to work with portions of the signal that contains no more than one musical event, for example to separate the different notes of a melody even for fast ones that can appear, for instance, in jazz solos.

Before extracting the features from these frames, a selection step is performed to retain only the frames which contain the steady state portion of the signal and exclude the ones which include silence, or transients. In fact, preliminary empirical results show that, given our feature set, common statistical classifiers (SVM, decision trees, Bayesian networks) fail to classify correctly the frames that contain transients. To do so, we first use a noise gate with a -13dB threshold to remove *silent* frames. To detect quickly transient frames, we use a simple onset/offset detection algorithm, presented in Figure 2, which computes the difference between the RMS values of the 10 first and last milliseconds of the signal, and applies a 6dB threshold on it.

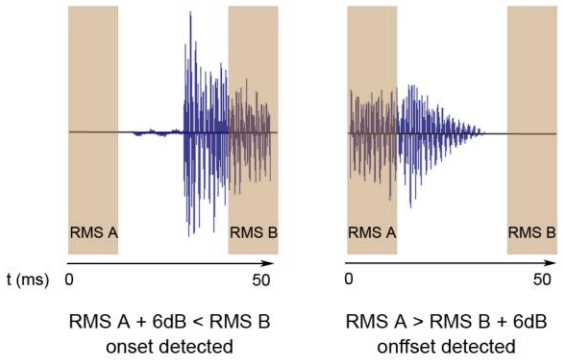

Fig. 2. Simple onset/offset detection procedure. The output of the algorithm is positive if there is a difference of 6dB or more between the two RMS values.

More sophisticated onset detection techniques such as frequency domain-based ones [4] can be used, but the proposed solution is fast and works well enough for our goals. Finally, the features are extracted from the remaining audio frames.

We finally train and test various classifiers on our database: a Support Vector Machine with linear, radial and polynomial kernels, a Bayesian network and a J48 tree. The best classifier turns out to be a Bayesian network classifier (Weka's *BayesNet* with a "GeneticSearch" algorithm) with default parameters.

3.3 Performance on Frame Classification

We train the classifier on one song, *The Days of Wine and Roses* (the longest song of the database), taken from the Godin guitar (*God*) subset, and test it on the six other songs. When we classify the selected audio frames (discarding silent frames and transients) with our feature set and the Bayesian network, we obtain an average F-measure of 0.87. This result is not sufficient for a robust, real-time classifier. In the next section we add an aggregation, or smoothing step to our method to further improve the classifier performance, following the approach in [2].

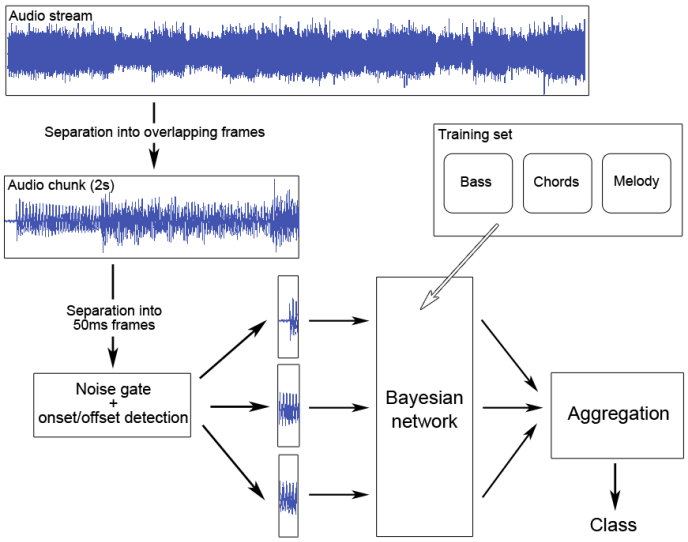

Fig. 3. General scheme of the classification algorithm.

3.4 Aggregation

In order to improve classification performance, we aggregate the results of individual frame classification within a given time window (called thereafter chunk) and apply a winner-takes-all strategy to identify the mode of the chunk. A typical chunk size is one bar at reasonable tempo (1*s* at 240 bpm, 4*s* at 60 bpm).

Since the Bayes classifier yields a probability of class membership, we can discard frames for which the class probability falls under a given threshold, which can be determined with a validation process, and use the winner-take-all strategy on the remaining ones.

4 Results

This section describes various evaluations of the classifier (including aggregation), highlighting the impact of using different guitars on classification robustness.

4.1 Evaluation on a one-song training set

First, we train the classifier on one single song, “The Days of Wine and Roses”, taken from the Godin guitar (*God*) subset. Then, we test it on the six other songs, for each guitar subset, with a 1.5s chunk duration (the duration of one 4/4 bar at 160 bpm). The results are displayed on Table 1.

Table 1. Classification performance obtained over six songs, for various guitar models.

Tested subset	God	Cort	Ovat	Nyl
Mean F-measure	0.96	0.941	0.854	0.839

For the guitar subsets *Cort* and *God*, the results are slightly better than the preliminary ones obtained with the Bayesian network without the aggregation step (.87 average F-measure). However, the classification results are poor for the *Ovat* and *Class* guitar subsets.

4.2 Evaluation with the use of larger training sets

To improve the performance, we *increase* the size of the training set: we train and evaluate the classifier with the leave-one-out procedure. Hence, each training set contains now six songs. To study the influence of the guitar type used for training and testing, we repeat this procedure for each guitar subset. The results are displayed on Table 2.

Table 2. Classification performance obtained with the leave-one-out procedure on the whole dataset. The first number is the minimum F-measure over the six tested songs, the second is the average F-measure.

Tested subset →	God	Cort	Ovat	Nyl
Training set : God	0.956 0.971	0.933 0.968	0.654 0.90	0.71 0.901
Training set : Cort	0.94.3 0.963	0.974 0. 984	0.753 0.94	0.922 0.972
Training set : Ovat	0.885 0.92	0.917 0.955	0.964 0.978	0.956 0.978
Training set : Nyl	0.92 0.943	0.961 0.975	0.961 0.975	0.981 0.992
Average F-measure over all training sets	0.949	0.971	0.948	0.961

These results show that while a larger training set increases the accuracy, the classification performance depends of the guitar used for training and testing: more specifically, the pairs *God/Cort* and *Ovat/Nyl* seem to give better results when used together (one for training and the other for testing). This can be explained by the fact that the guitars used to record Ovat and Nyl subsets produce more high-frequency content than the other ones: a feature such as spectral spread is sensitive to timbre.

4.3 Evaluation with a mixed training set

In order to make the classifier more independent of the guitar type, or more generally of timbral variations, we pick tracks from *each of the four subsets* to build a new training set. We use the recordings of *The Days of Wine and Roses* and *Ladybird* from

each subset to train the classifier, and test the performance on the six remaining tracks. Results are shown on Table 3.

Table 3. Classification performance obtained with the use of a mixed training set. We compare the minimal F-measures over the four guitars in order to evaluate the timbral sensitivity of the classifier.

Tested subset →	God	Cort	Ovat	Nyl	Min. F-measure
Bluesette	0.971	0.988	0.989	0.995	0.971
Nardis	0.941	0.966	0.973	0.99	0.941
Ornithology	0.99	0.988	0.99	0.999	0.988
Solar	0.977	0.965	0.985	0.997	0.965
Tune Up	0.968	0.984	0.962	0.952	0.952
Min.F-measure per tested subset	0.968	0.978	0.98	0.987	0.968

Here, we can see that the use of a mixed training set, containing two songs (or a total 31 minutes of audio), increases the overall performance. We evaluated the classifier with larger training sets, but larger sets do not increase classification accuracy in a significant way. This last training set will be used in the rest of this article.

4.4 Influence of the analysis window length

Since the algorithm includes an aggregation step, we can assume that the accuracy of the classifier depends on the length of the analyzed audio chunks. Figure 4 displays the performance of classification, obtained over the five tracks which are not included in the training set, for various analysis windows. As a comparison, we added, for each guitar subset, the F-measures obtained without performing the aggregation over the 50ms frames.

We can see that the classification improves when increasing the analysis window length, reaching a plateau at about .98.

4.5 Real-time

Since the algorithm consists in feature extraction and simple Bayesian classification, the overall complexity of the algorithm is linear with the analyzed audio window length (other computation such as the aggregation is negligible). The average classification CPU is 2% of real-time, with a Java implementation running on an Intel i7 2.67GHz quad-core, Windows laptop (eg. the experienced latency obtained for the analysis of a 4/4 bar at 120 bpm is 20ms). This clearly enables interactive musical applications on commonly available hardware.

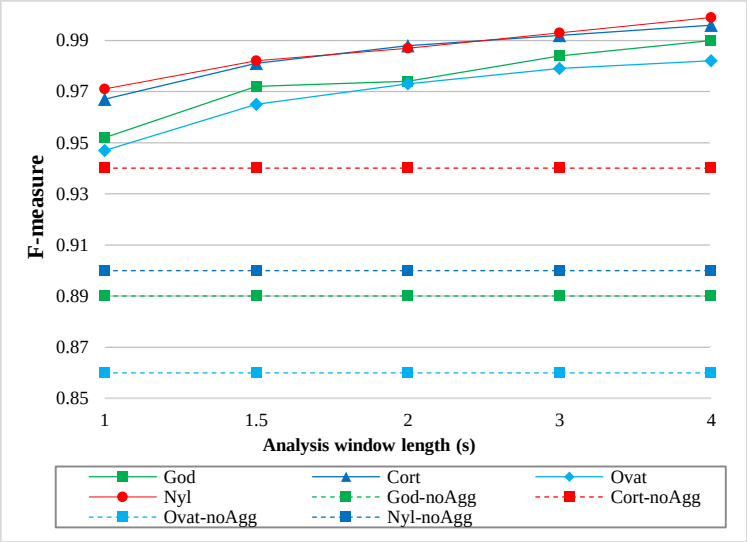

Fig. 4. Improvement of classification performance obtained with different analysis window lengths. The series followed by the mention *noAgg (dotted lines)* show the results obtained without the aggregation step.

4.6 Comparison with MIDI-based classification

We compared our algorithm with the performance of a MIDI-based mode classifier described in [10], using the Godin guitar subset (this guitar has a MIDI output). The MIDI-based classifier trains a Support Vector Machine classifier on 8 MIDI features, related to pitch, duration, velocity, and more advanced ones, aggregated over one bar. This classifier is been trained on the song *Bluesette*, and tested on the six other songs. In order to work with the same experimental settings, we adapted the analysis frame of our audio-based algorithm on each song, to match the length of a bar. The results are displayed in Tables 4 and 5.

Table 4. Results of the MIDI-based SVM classifier

Classified as →	Bass	Chords	Melody	F-measure
Bass	1314	6	2	0.98
Chords	24	1294	13	0.97
Melody	1	12	1318	0.99

Table 5. Classification results obtained with our classifier

Classified as →	Bass	Chords	Melody	F-measure
Bass	1118	1	3	0.947
Chords	85	1017	11	0.936
Melody	25	31	1065	0.968

The results we obtain are still reasonable, but weaker than with the preceding training sets. This is due to the fact that audio analysis requires larger training sets than MIDI to reach the same performance. To illustrate this point, we increase slightly our training set and train our classifier with two songs: *Bluesette* and *The Days of Wine and Roses*. We repeat the testing procedure on the five remaining tracks. The confusion matrix is displayed on Table 6.

Table 6. Classification results obtained with our classifier, with a larger training set.

Classified as →	Bass	Chords	Melody	F-measure
Bass	811	30	10	0.965
Chords	0	852	3	0.969
Melody	18	21	817	0.969

These results show that our method provides results which are comparable to the ones obtained with the MIDI output of the guitar. This result enables us to integrate our algorithm in actual interactive live applications, without any MIDI support.

5 Interactive applications

We describe two examples of applications of our mode classifier. VirtualBand, which implementation is discussed in [10], enables a musician to control virtual instruments which interact with him in real time, while following a harmonic grid. A VirtualBand's applications, *reflexive loop pedals*, use our mode classifier to analyze the audio input. Two virtual instruments are instantiated by the system: *bass* and *chords*. In a typical interaction scenario, the musician starts to play chord, which are recognized as such and recorded by the system. Then, the musician starts playing a walking bass line. As a response, the *chords* virtual instrument plays back recorded chord audio chunks, transposing them if necessary with a pitch shifting algorithm, to fit the grid. The musician can then decide to play some melodic lines. In that case, the *bass* and *chords* virtual instruments both play along. A detailed version of this example is given in [10]. This interactive application allows the user to control the musical process while not using control devices such as loop pedals, that would interfere with his creative flow. Hybrid modes, such as the "Bossa nova" mode (see next Section) can be added to this setup to allow more sophisticated interactions, thanks to the addition of a new virtual instrument.

Another application, included in VirtualBand's features, is to automatically process the input sound according to the playing mode. Various audio effect chains can be applied to the audio input, their selection depending on the currently played mode. For instance, the system can add a specific reverberation effect when the musician is playing melody, tube distortion when chords are detected, and, say, apply dynamic compression and enhance the low end of the sound when the instrumentalist is playing bass. The effect chain is applied with a latency corresponding to the chunk size needed to perform mode classification (about 1 sec minimum).

6 Discussion

We have shown that YIN features, that represent half of our feature set, are efficient to classify guitar playing modes: our classifier is accurate, robust to variations in guitar type, and able to cope with real-time computational constraints, thanks to a small feature set. We also showed that although the accuracy depends on the size of the analyzed audio, this classifier can be used with realistic window sizes. Three points can be improved, to further extend its applicability.

6.1 Features

The method also raises issues when dealing with long chord decays (say, more than 5s), when only one note keeps sounding. This case falls off the boundaries of our algorithm and feature set. One solution would be to add a robust onset detector to our algorithm, and restrict the mode computation on the first seconds that follow an onset (we did no implement such a solution).

Another limitation comes from the feature set: we work with a feature set that answers a specific problem, but it may not be efficient to distinguish efficiently yet other playing modes, such as strums or octaves. The algorithm is also somewhat specific to the guitar: the YIN inharmonicity factor may not behave the same with less harmonic instruments, such as the piano.

6.2 Hybrid playing modes

In our method, we perform an aggregation step because the frame classifier alone is not accurate enough. Nevertheless, it provides a good hint about the rate of chords, melody and bass, within audio chunks that contain a mixture of different playing modes. For instance, we can consider an extra "Bossa nova" playing mode which consists in alternative bass/chords patterns. In order to recognize such a mode, we add an extra rule to the aggregation step of the algorithm: before applying the winner-takes-all strategy to our frames classes, we compute the weight of each class, without taking the class probabilities into account, and we express it in absolute percentage. Then, we consider the bass and chords weights: if they are both greater than, say, 20% and lower than 80%, then we can consider that the chunk belongs to the "Bossa nova" class. Such a rule could be also implemented in a classifier, so that the process is entirely automatic. An example of such a hybrid mode is displayed in figure 5.

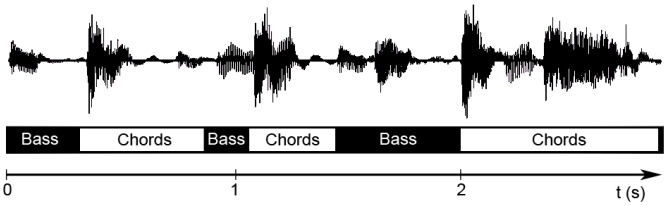

Fig. 5. Identification of bass and chord parts in a Bossa nova guitar audio signal

Although the frame classifier does not provide an accurate weight for each class within a chunk, the ability to detect when the musician is performing this hybrid playing mode brings new possibilities for building interactive applications. This pattern is correctly detected, however it represents only a particular case of the Bossa nova playing mode, in which bass and chords do not overlap. In the (frequent) case when they overlap, the classifier performance drops sharply.

6.3 Absolute aggregation vs. time series

In this paper, we use the simple winner-take-all strategy to aggregate the 50ms frames over the entire analysis window. This method does not take into account the time-series nature of a musical audio signal. For instance, guitar players sometimes use low-pitched notes in their melodic improvisation, and conversely, play walking bass with high-pitched notes. With our window-based scheme, the classifier uses the YIN pitch as a strong hint to distinguish the melody from the bass. As a consequence, the user might be surprised by some results for those notes with intermediary pitches (e.g. in the range C3-E3) also, since there is no high-level musical analysis of the currently played phrase. The evolution and continuity between the different features values extracted within an audio chunk could be evaluated over time, leading to a smarter way to process these frames. We assume that a classifier that exploits such knowledge would be more accurate, and also could efficiently identify more sophisticated new playing modes such as arpeggios, muted notes strumming and, in general, playing modes based on longer temporal patterns.

7 Conclusion

We have presented a simple, fast and accurate method to classify basic playing modes for the guitar from its audio signal. We limited our work to the study of the guitar, focusing on a timbre-independent classifier. Further studies with other instruments such as the piano or synthetic sounds are needed to assess the generality of our approach.

Our classifier works efficiently for simple playing modes, such as *chords* or *bass*. Future designs of the algorithm, in particular taking into account the continuity between frame analysis, will be able to distinguish more complex playing modes, such as the Bossa nova bass/chord mixing discussed earlier.

Automatic playing mode classification brings a lot of potential for designing smarter augmented instruments. Interestingly, developing subtler and subtler playing modes classifiers, from polyphony-based detection as we presented here to the identification of player-specific patterns (Montgomery's octaves, Benson's licks or Van Halen's fast harmonic arpeggios), infringes on the emerging domain of style modeling.

Acknowledgments

This research is conducted within the Flow Machines project which received funding from the European Research Council under the European Union's Seventh Framework Programme (FP/2007-2013) / ERC Grant Agreement n. 291156.

8 References

1. Abesser J., Lukashevich H.M., and Schuller G.: Feature-Based Extraction of Plucking and Expression Styles of the Electric Bass Guitar. In: Proc. International Conference on Acoustics, Speech and Signal Processing, pp. 2290--2293, Dallas (2010).
2. Abesser J.: Automatic String Detection for Bass Guitar and Electric Guitar. In: Proc. of the 9th International Symposium on Computer Music Modelling and Retrieval, pp. 567--582, London (2012).
3. De Cheveigné A., Kawahara H.: YIN, a Fundamental Frequency Estimator for Speech and Music. In: Journal of the Acoustical Society of America, vol. 111, no. 4, pp. 1917--1931, 2002.
4. Dixon S.: Onset Detection Revisited. In: Proc. of the 9th International Conference on Digital Audio Effects, pp. 113--137, Montreal (2006).
5. Hartquist J.: Real-Time Musical Analysis of Polyphonic Guitar Audio. Ph.D. Thesis, California Polytechnic State University (2012).
6. Klapuri A., Davy M., Signal Processing Methods for Music Transcription. Springer, (2000).
7. Lachambre H., André-Obrecht R., and Pinquier J.: Monophony vs Polyphony: a New Method Based on Weibull Bivariate Models. In: Proc. of the 7th International Workshop on Content-Based Multimedia Indexing, pp. 68--72, Chania (2009).
8. Lähdeoja O., Reboursière L., Drugman T., Dupont S., Picard-Limpens C., Riche N. : Détection des Techniques de Jeu de la Guitare. In: Actes des Journées d'Informatique Musicale, pp. 47--53, Mons (2012).
9. Özaslan T.H, Guaus E., Palacios E., Arcos J.L.: Attack Based Articulation Analysis of Nylon String Guitar. In: Proc. of the 7th International Symposium on Computer Music Modeling and Retrieval, pp. 285--297, Malaga (2010).
10. Pachet F., Roy P., Moreira J., D'Inverno M.: Reflexive Loopers for Solo Musical Improvisation. In: Proc. of International Conference on Human Factors in Computing Systems (CHI), pp. 2205--2208, Paris (2013). Best paper honorable mention award.
11. Reboursière L, Lähdeoja O., Drugman T., Dupont S., Picard-Limpens C., Riche N.: Left and Right-Hand Guitar Playing Techniques Detection. In: Proc. of New Interfaces for Musical Expression, pp. 30--34, Ann Arbor (2012).
12. Ryynänen M.P., Klapuri A.P.: Automatic Transcription of Melody, Bass Line, and Chords in Polyphonic Music. In: Computer Music Journal, vol.32, no.3, pp. 72--86, MIT Press (2008).
13. Stowell D., Plumbley M.D.: Delayed Decision-Making in Real-Time Beatbox Percussion Classification. In: Journal of New Music Research, vol. 39, no. 3, pp. 203--213, Routledge (2010).
14. Sinyor E., Mckay C., Mcennis D., Fujinaga I.: Beatbox Classification Using ACE. In: Proc. of the 6th International Conference on Music Information Retrieval, pp. 672--675, London (2005).
15. Stein M., Abesser J., Dittmar C., Schuller G.: Automatic Detection of Audio Effects in Guitar and Bass Recordings. In: Proc. of the 128th Audio Engineering Society Convention, London (2010).
16. Tindale A., Kapur A., Tzanetakis G., Fujinaga I.: Retrieval of Percussion Gestures Using Timbre Classification Techniques. In: Proc. of the 5th International Conference on Music Information Retrieval, pp. 541--544, Barcelona (2004).
17. Guyon, I., Elisseef, A.: An Introduction to Variable and Feature Selection. In: The Journal of Machine Learning Research, vol. 3, pp. 1157--1182. JMLR.org, (2003).
18. Fohl W., Turkalj I., Meisel A.: A Feature Relevance Study for Guitar Tone Classification. In: Proc. of the 13th International Study for Music Information Retrieval Conference, Porto (2012).
19. Barbancho I., Tardon L.J., Sammartino S., Barbancho A.M.: Inharmonicity-Based Method for the Automatic Generation of Guitar Tablature. In: IEEE Transactions on Audio, Speech, and Language Processing, vol. 20, no 6, p. 1857--1868, (2012).

20. Holmes G., Donkin A., Witten I.H.: Weka: A Machine Learning Workbench. In: IEEE Proc. of the Second Australian and New Zealand Conference on Intelligent Information Systems, pp. 357--361, (1994).