Voice-controlled plucked bass guitar through two synthesis techniques

Jordi Janer
IUA-MTG
Universitat Pompeu Fabra
Barcelona
jjaner@iua.upf.es

ABSTRACT

In this paper we present an example of the use of the singing voice as a controller for digital music synthesis. The analysis of the voice with spectral processing techniques, derived from the Short-Time Fourier Transform, provides ways of determining a performer's vocal intentions. We demonstrate a prototype, in which the extracted vocal features drive the synthesis of a plucked bass guitar. The sound synthesis stage includes two different synthesis techniques, Physical Models and Spectral Morph.

Keywords

Singing voice, musical controller, sound synthesis, spectral processing.

1. INTRODUCTION

Digital music synthesis allows a wide range of sounds and permits a high degree of control over the sound generation process. It overcomes physical (in acoustic instruments) or electronic (analog synthesizer) limitations in terms of control. The flexibility of control has led to the emergence of numerous musical controllers, more innovative than the traditional keyboard approach, ranging from accelerometers to video cameras. The captured data is then converted to some protocol that controls the synthesis engine. Traditionally, this protocol was MIDI, but Open Sound Control (OSC) is gaining more and more acceptance in the Computer Music community, as well as in the industry.

Singing voice qualities have been exploited in the history of music since ancient times. The emotion transmitted by an opera singer is indubitable, probably due to the fact that the voice is *per se* an organic musical instrument. It is not the aim of this paper to study in detail the characteristics and qualities of the singing voice as a musical instrument. Rather, we address the high degree of expression and the nuances of the singing voice in order to exploit it as a musical controller. Music Technology has tackled the singing voice mainly from the analysis / synthesis perspective, putting efforts in "inventing" a computer that sings as a human. Another topic, in which the singing voice is involved, is *score following* systems [12].

1.1 Voice as a Controller

In the work presented here, the singing voice acts as a controller. The system examines the characteristics of the captured acoustical signal, which at its turn, drives the parameters of a synthesis engine. Other approaches of "sound-controlled sound synthesis" are found in the literature. Tristan Jehan [8] presents a system developed at MIT which uses the timbre characteristics of a continuous input stream to recreate the output with characteristics of another pre-analyzed instrument. In another approach by Miller Puckette et al. [13], the goal is to map a low-parametric timbre space of an input audio stream onto a pre- analyzed output audio stream in real-time. Also related to this work, we should include *PitchToMidi* systems, which were first introduced in the 80's as hardware devices. For our purposes, though, MIDI presents mainly two limitations. First, it is an event-based protocol, and the voice -as many other instruments- varies its sounds in a continuous manner. Second, the available bandwidth offers an insufficient time resolution.

The plucked bass guitar is, in terms of timbre complexity, simpler than other solo instruments such as a violin or a trumpet. This *simplicity* let us focus on basic control aspects. Furthermore, we want to stress that the goal of the presented work is not to build an accurate bass guitar synthesizer, but rather to explore the possibilities of the voice driven synthesis.

2. VOICE FEATURE EXTRACTION

The first step before using the voice as a controller is to decide which features we want to extract. It is obvious that even a naive user can control efficiently the phonatory system, producing a wide range of sounds. Actually, the simple action of speaking involves continuous variations of the breathing pressure, vocal folds tension or tongue position, to name a few. The chosen features include basic attributes, such as pitch and energy, in addition to other timbre descriptors.

2.1 Voice Processing techniques

Several techniques have been developed over the years to study the human voice. Most research is related to the field of speech processing, but the study of the singing voice

has brought also additional approaches. Basically, all these methods, though, rely on the source/filter approach, in which the phonatory system is seen as a coupled excitation (modulated air-flow) and a resonator (vocal tract cavity).

In this work we use a Phase-Vocoder based method for voice analysis and also for the spectral morph synthesis. The Phase Vocoder was first developed by Flanagan [4] in 1966 at Bell Laboratories, and was used for speech processing. Later, in its FFT form was brought to musical applications by J.A. Moore and M. Dolson. M. Puckette proposed the Phase-Locked Vocoder [11], which introduces phase- locking between adjacent pairs of FFT channels. A further step in the Phase-vocoder development is the new algorithm proposed by J. Laroche in [10]. This technique allows direct manipulation of the signal in the frequency domain. Some of the application are pitch-shifting, chorusing and harmonizing. The underlying idea behind the algorithm is to identify peaks in the Sort-Time Fourier Transform, and translate them to new arbitrary frequencies of the spectrum. Here, the spectrum is divided in several regions around peaks.

This technique was integrated in the *Spectral Peak Processing* (SPP) framework [1] by Bonada and Loscos. It performs a frame based spectral analysis of the audio, giving as output of the STFT, the harmonic peaks and the pitch, as depicted in figure 1. For the pitch detection, we used the technique developed by Cano et al. described in [2]. Basically, the SPP considers the spectrum as a set of regions, each of which belongs to one harmonic peaks and its surroundings. The main goal of such technique is to preserve the local convolution of the analysis window after transposition and equalization transformations. Common transformations include pitch-shift and equalization (*timbre modification*).

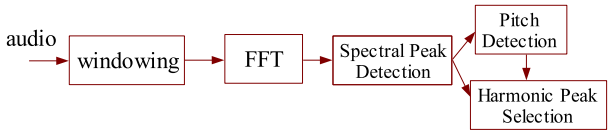

Figure 1: SPP Analysis process.

2.2 Set of extracted features

Essentially, vocal gestures can be of three different kind in the voice production mechanism, depending on whether its origin is either in the breathing system (subglottal pressure), glottis (glottis settings), or in the vocal tract (vocal tract shape). In this approach, we propose the following classification based on control aspects: *Excitation*, *Vocal Tract*, *Voice Quality* and *Context*.

A more detailed description of the feature extraction algorithms, which derive from the *SPP Analysis*, appears in an article by the author [7]. *Excitation* descriptors are elemental for the user, since they are related to the instantaneous sung energy and fundamental frequency. In a further step, we find the voice colour or voice timbre. For voiced sounds, the timbre is associated to a particular vowel. We assume that a vowel can approximately determined by its two first formant frequencies. A very simple algorithm based on spectral centroid, will estimate these frequencies approximately. In addition to the introduced features, two algorithms estimating *Voice Quality* were developed. Although these two algorithms need further research, it is a good start-point for controlling other aspects of the output sound. Finally, we include the *Context* features, which includes the "Attack unvoiceness" descriptor. Assuming that a note consists of a pitched sound, this descriptor attempts to determine whether the attack of the note was unvoiced or not. The initial motivation for this descriptor is to use it for determining the harshness of the synthesized note's attack. In our case of a bass guitar, it might be related to the differences between a soft fingered and a sharp slap electric bass.

3. PLUCKED-BASS GUITAR SYNTHESIS

The sound of a plucked bass guitar consists of impulsive notes that decay exponentially in time. Due to the long note decay time, bass players usually damp the vibrating string with the finger before plucking a new note. In the previous section we have proposed a set of attributes that derive from a spectral analysis of the voice signal, which delivers information at a given frame rate. Making an analogy, we could handle this information as it would come from other multi-parametric sensors such as a body- gesture device that sends tracking information continuously. On the other hand, as pointed out in the literature [5], the design of digital musical instruments involves a very important layer, the *Mapping*. We propose a general model valid for different instruments, as well as different synthesis techniques. Regarding the synthesis process, in the system here implemented we aimed to compare two dissimilar techniques, Physical Models and Spectral Morph.

3.1 Mapping issues

Once we have the input parameters available, the next task is to set up a meaningful strategy to link these parameters to controls of the synthesis algorithms. This action is commonly known as *mapping*. Although, here we present the synthesis of a single instrument, the bass guitar, it is part of a more general research on voice-controlled virtual instruments. In this sense, we propose a general model of mapping that can deal with different instruments and different synthesis techniques. The figure 2 shows the mapping process in which a first layer deals with the synthesized instrument, and a second layer that is connected to the synthesis engine. In the table 1, we justify this concept with some examples.

Instrument dependency	
Pitch	Tessitura (bass vs. violin) Pitch deviations (violin vs. piano)
Volume	Continuous (flute) Impulsive (plucked guitar)
Timbre variations	Instantaneous (piano attack) Continuous (violin)
Synthesis technique dependency	
Simple oscillator	frequency and amplitude
Physical Models	Length, Excitation force, Air-flow pressure, etc.
Spectral Models	Sinusoidal partials amplitude and frequency, etc.

Table 1: The Mapping layer consists of two sub-layers, which adapt the input parameters depending on the instrument sound and the synthesis technique.

In our case, the *instrument dependent* mapping is unique, since only one instrument is considered. From the *Energy*

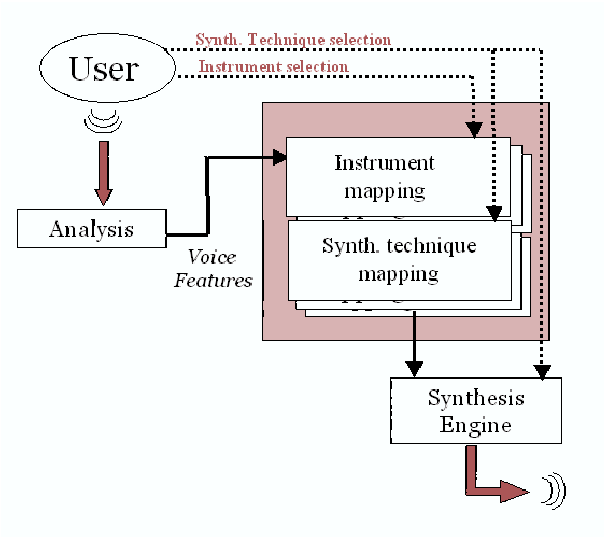

Figure 2: The mapping of the voice features is a two-step process: Instrument dependant and synthesis technique dependant

feature, we defined a note onset signal that triggers a note depending on the energy's derivative. The *Pitch* feature is transposed one octave lower and passed as continuous parameter, thus allowing pitch deviations. A quantization function for simulating the influence of the frets is foreseen, but not yet integrated in the current implementation.

3.2 Adapting the Karplus-Strong String Algorithm

This algorithm was invented by Kevin Karplus and Alex Strong, and was published in the Computer Music Journal in 1983 [9]. The success of such algorithm lies in its low-computational cost, which allowed at that time to synthesize in real-time a plucked-guitar with a substantial quality with cheap processors. It is worth to mention that in 1983 the first PCs were introduced; and the digital music synthesis in real-time required large and expensive computers only found in big research centres.

Although we have mentioned here Physical Models, originally this algorithm was based on the *Wavetable Synthesis* technique, which repeats number of samples generating a periodic signal. Instead, the Karplus-Strong algorithm introduces a variation by averaging two successive samples $Y_t = (Y_{t-N} + Y_{t-N-1})/2$, and writing the resulting sample in the wavetable. This can be seen, thus, as a delay line with length N. It simulates a rigidly terminated string with losses, generating a periodic sound with decay. The delay line is initialized with random values (+A/-A) for simulating the *plucking* action. The pitch is determined by the length of the delay line. At CCRMA, researchers experimented simultaneously with the Karplus-Strong algorithm, proposing some extensions, and analyzing it in terms of a digital filter. The extensions developed by D. Jaffe and J.O. Smith [6] overcame several limitations of the original algorithm.

We adapted this algorithm in order to be controllable by the voice, as depicted in the figure 3. Unlike the original algorithm, in which the delay line is filled by noise, we feed the Karplus-Strong delay line with the transient (attack) of

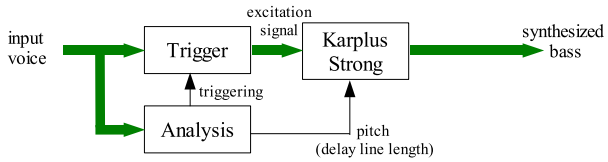

Figure 3: Bass synthesis with Physical Modelling. The input voice is used as excitation signal for the Karplus-Strong algorithm.

the sung note. Good results were achieved by attacking a note with a short impulsive consonant. In our bass synthesizer system, the implementation of the Karplus-Strong Algorithm is taken from the STK Library [3]. From the first mapping sublayer, we get the triggering signal, which is related to the energy envelope of the user's voice, and the estimated pitch. When the trigger is active, N samples of the input voice passes through and fill the delay line. The actual size of the delay line (N) is determined by the estimated pitch.

During the design process, we noticed that sample continuation problems appeared in form of "clicks" in the output waveform, when changing from one note to another. This is caused while a note is still fading out, and a new excitation signal is fed into the delay line. In order to minimize this effect, we introduce a second *string* in our model. The triggering stage send the excitation signal alternatively to one of the two strings, sounding more natural.

3.3 Spectral Morph algorithm

Our Spectral Model approach combines a sample-based synthesis with transformations, based in the Spectral Peak Processing (see section 2.1). A particularity of our sample-based algorithm is that it works in the frequency domain. Basically, depending on the input voice's parameters, a sound template track is selected from the database. Each template track contains all spectral frames from a single note. Then, we read sequentially the spectral frames and apply some transformations. Finally, the processed spectral frames are converted to time domain through the inverse Fourier transform.

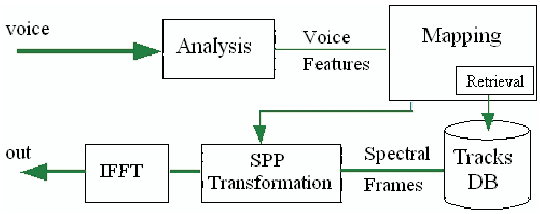

Figure 4: The voice features select a track from the database. The stored track's spectral frames are transformed and finally converted to the time domain.

For our bass synthesizer implementation, we set up a very small database consisting of 12 template tracks (one note sample), considering two playing techniques. The tracks are analyzed off-line in the spectral domain, and stored in the database in form of binary files containing spectral data (complex spectrum, harmonic peaks, estimated pitch, etc.).

All tracks have been labelled by hand according to its characteristics. Currently, only three features were annotated: *Pitch*, *Dynamics* and*Attack Type*. The pitch values are specified in Hz, *Dynamics* and *Attack Type* range is [0..1]. In the case of Dynamics, 0 corresponds to a *pp* sound, and 1 to a *ff*. The attack type is instrument dependant. Concerning bass sounds, we decided to classify two types of sounds depending on the plucking technique soft fingered or slap.

Regarding the mapping for the spectral morph model algorithm, a retrieval method calculates the minimum Euclidean distance between the Input Vector (Pitch, Loudness and Attack Unvoiceness) and the database elements. In a further step, we start reading the spectral frame of the selected track. Since we are dealing with a very small database, few combination of loudness and pitches are available. There are two types of transformations: transposition and gain.

Another factor that must be taken into account is the timing. The pre-analyzed template tracks have a certain duration, a certain number of spectral frames. In our system, though, the performer's voice controls the synthesized sound. Hence, is the input voice which decides the output duration. We mark manually a *sustain* point (frame). After a note is triggered, we read frame by frame until the *sustain* frame is reached. Then, this frame is read repeatedly, considering phase continuation, until the user releases" the sung note. During the sustain, pitch bend can occur. and only one note sample is read from the database at a given time.

3.4 Implementation description

The proposed system was implemented as a stand-alone C++ application running in real time on a off-the-shelf computer. We used the CLAM [1] development framework, and the *PortAudio*[2] library for the audio interface along with ASIO[3] drivers. The internal algorithms parameters were: $windowsize = 2048$, and $hopsize = 256$. The real minimum latency of our set up (virtual audio wire) was $11ms$. For the Physical Modelling approach, we observed a latency of 26 ms. In the Spectral Morph technique, the latency raises up to $34ms$. Note that the latency in the latter case is increased due to the FFT and IFFT process, in which a latency of half window size is unavoidable.

4. CONCLUSIONS

The achieved results showed that, in terms of computation requirements, such a system can be utilized as a virtual instrument in a real-time performance situation. Nevertheless, this first approach is still limited in terms of musical control, and therefore needs further development. Among the two synthesis techniques integrated, *Physical Models* has in this case much less computational and memory cost than the *Spectral Morph* approach. The quality of the synthesized sound, a plucked bass guitar, is in both techniques acceptable. However, since we plan to extend this system to other musical instruments, Physical Modelling technique requires for each instrument a completely different algorithm, which implies extra design effort. Instead, the Spectral Morph approach is more scalable, in which adding new instruments requires only to provide a set of recorded note samples to the database, and to adapt the mapping layer.

[1]http://www.iua.upf.es/clam
[2]http://www.portaudio.com/
[3]Audio Stream I/O (ASIO) is trademark of Steinberg, GmbH.

5. ACKNOWLEDGMENTS

This research has been partially funded by the EU-FP6-IST- 507913 project SemanticHIFI.

6. REFERENCES

[1] J. Bonada and A. Loscos. Sample-based singing voice synthesizer by spectral concatenation. In *Proceedings of Stockholm Music Acoustics Conference 2003*, Stockholm, Sweden, 2003.

[2] P. Cano. Fundamental frequency estimation in the SMS analysis. In *Proceedings of COST G6 Conference on Digital Audio Effects 1998*, Barcelona, 1998.

[3] P. Cook and G. Scavone. *The Synthesis Toolkit (STK)*. http://ccrma-www.stanford.edu/software/stk.

[4] J. Flanagan and R. Golden. Phase vocoder. *Bell Systems Technology Journal*, 45:1493–1509, 1966.

[5] A. Hunt, M. Wanderley, and M. Paradis. The importance of mapping in electronic instruments design. In *Proceedings of the Conference on New Instruments for Musical Expression NIME*, Dublin, Ireland, 2002.

[6] D. Jaffe and J. Smith. Extensions on the Karplus-Strong plucked-string algorithm. *Computer Music Journal*, 7(2):56–69, 1983.

[7] J. Janer. Feature extraction for voice- driven synthesis. In *118th AES Convention*, Barcelona, 2005.

[8] T. Jehan. *Perceptual Synthesis Engine: An Audio-Driven Timbre Generator*, 2001.

[9] K. Karplus and A. Strong. Digital synthesis of plucked-string and drum timbres. *Computer Music Journal*, 7(2):43–55, 1983.

[10] J. Laroche and M. Dolson. New Phase-vocoder techniques for pitch-shifting, harmonizing and other exotic effects. In *Proceedings of IEEE Workshop on Applications of Signal Processing to Audio and Acoustics*, New Paltz, New York, 1999.

[11] M. Puckette. Phase-locked vocoder. In *Proceedings of IEEE Workshop on Applications of Signal Processing to Audio and Acoustics*, Mohonk, New York, 1995.

[12] M. Puckette. Score folowing using the sung voice. In *Proceedings, International Computer Music Conference*, San Francisco, 1995.

[13] M. Puckette. Low-dimensional parameter mapping using spectral envelopes. In *Proceedings, International Computer Music Conference*, Miami, 2004.