
	SemanticHIFI <i>IST-507913</i>
Public Report of WP5 “Performing” Covering period: December 2003 – August 2006 Report version: 2.1 Report preparation date: November 2006 Writers: Sergi Jordà, Jordi Janer, Alex Loscos (UPF), Diemo Schwarz (IRCAM) Control: Hugues Vinet, Francis Rousseaux, IRCAM Classification: Public Contract start date: December, 1st 2003 Duration: 32 months Project co-ordinator: Hugues Vinet, IRCAM Involved Partners: UPF (WP Co-ordinator), IRCAM, Sony CSL	
 Information Society Technologies	Project funded by the European Community under the "Information Society Technology" Programme

Table of contents

1	WP Overview	4
1.1	Objectives	4
1.2	Partners' roles.....	4
1.3	WP contribution to the project	4
1.4	Synthesis of main achievements	5
2	WP Results and Achievements.....	6
2.1	WP5.1 Voice Transformation.....	6
2.1.1	Functional description	6
2.1.2	Method description	6
2.1.3	Position over state-of-the-art.....	7
2.1.4	Benchmarks	10
2.1.5	Implementation	11
2.1.6	Dissemination materials	12
2.1.6.1	Scientific publications	12
2.1.6.2	Public presentations	12
2.1.6.3	Products.....	12
2.1.6.4	Related PhDs.....	13
2.2	WP5.2 Voice Instrumentizer	14
2.2.1	Functional description	14
2.2.2	Method description	14
2.2.3	Position over state-of-the-art.....	14
2.2.4	Benchmarks	15
2.2.5	Implementation	16
2.2.6	Dissemination materials	19
2.2.6.1	Scientific publications	19
2.2.6.2	Public presentations	19
2.2.6.3	Related PhDs.....	19
2.3	WP5.3 DJ Voice-Controller.....	20
2.3.1	Functional description	20
2.3.2	Method description	20
2.3.3	Position over state-of-the-art.....	21
2.3.4	Benchmarks.....	22
2.3.5	Implementation	23
2.3.6	Dissemination materials	24
2.3.6.1	Scientific publications	24
2.3.6.2	Other scientific dissemination actions.....	24
2.4	WP5.4 Rhythm transformation	25
2.4.1	Functional description	25
2.4.2	Method description	26
2.4.3	Position over state-of-the-art.....	26
2.4.4	Benchmarks	27
2.4.5	Implementation	29
2.4.6	Dissemination materials	31
2.4.6.1	Scientific publications	31
2.4.6.2	Public presentations	31
2.4.6.3	Related PhDs.....	31
2.5	Player for the HiFi System.....	32
2.5.1	Functional description	32
2.5.2	Method description	32

2.5.3	Position over state-of-the-art	32
2.5.4	Benchmarks	32
2.5.5	Implementation	32
2.5.5.1	Input:.....	32
2.5.5.2	Plugins:.....	34
2.5.5.3	Player Control	34
2.5.5.4	Filenames	34
2.5.5.5	Command Specification	34
2.5.5.6	Callback Messages from the Player.....	36
2.5.5.7	Future Features of the Player	36
2.5.6	Dissemination materials	37
2.6	Score Follower	38
2.6.1	Functional description	38
2.6.2	Position over state-of-the-art.....	38
2.6.3	Implementation	39
2.6.4	Dissemination materials	41
2.6.4.1	Scientific publications	41
2.6.4.2	Related PhDs.....	42

1 WP Overview

1.1 Objectives

This workpackage encompasses different applications that provide ways for the users of the final prototypes, to become active listeners, allowing them to perform while listening and to interact and modify the music being played on the system, by means of simple instruments or devices, such as a microphone.

The goals of the workpackage include performing music while listening with, home devices, in an edutainment context (instrumental performance learning) or in a creative context (DJ Tools).

1.2 Partners' roles

The three partners involved in this WP study and bring different and complementary approaches to this “simple performing” objectives:

- UPF focuses on the interaction with songs in the database, by means of the singing voice in what could be generalized as an ‘extended karaoke’, and in the musical conduction via simple interfaces (such as the remote HIFI controller).
- Ircam-RTA focuses on score-following and different spectral processing real-time algorithms and tools.
- Sony CSL focuses on functions by which the user can interact with a sound database. This interaction includes querying the database for similar timbre music or similar musical phrases.

1.3 WP contribution to the project

With the developments provided by the WP5 *Performing*, the project prototypes integrate novel tools for musical interaction. The contributions of this work package are an add-on to the final projects results, extending its initial features. It increases user engagement when learning about the *SemanticHiFi*.

All of these workpackage prototypes are being entirely developed for the SHF project; they constitute the continuation of previous research on voice transformation and on time-stretching algorithms and score-following. The development stage of each of the aforementioned three distinct research lines is further described in the following sections.

1.4 Synthesis of main achievements

Several final prototypes for the “HIFI system” as well as for the “Authoring Tool” have been developed and documented. The focus of the work was on user requirements, use cases, user interfaces and the software architecture of the two systems. Some of these prototypes are already in the form of separate plug-in applications that will run on either one of the two hosts systems (“HIFI system” or “Authoring Tool”).

For the former (HIFI system), an audio playback module with tempo and swing transformation has been developed. The latter (Authoring Tool) integrates a number of VST plugins, ranging from voice conversion to rhythm transformation. It includes *voice-processing applications* (that allow the user to sing on top of a soundtrack while modifying his or her voice in real-time), *voice-controlled applications* (that allow the user to control additional instruments such as bass or drums by means of his or her voice), and more general *sound processing applications* using time-stretching (i.e. modify the speed at which a soundtrack is played without affecting its pitch).

In addition, the Score Following Player creates an easy to use and robust automatic accompaniment application accepting monophonic audio and MIDI input from the performer.

More detailed information about these functional modules is to be found in each of the following subworkpackage sections.

2 WP Results and Achievements

In this section, the following components and sub-workpackages are discussed:

- Voice Transformation (WP5.1)
- Voice Instrumentizer (WP5.2)
- DJ Voice-Controller (WP5.3)
- Rhythm transformation (WP5.4)
- Player for the HiFi System
- IRCAM-ATR Score Follower (WP5.5)

2.1 WP5.1 Voice Transformation

Responsible partner: UPF

2.1.1 Functional description

Voice transformation has been an area of exploration probably ever since Lester Polfus (best know as Les Paul) started playing with tape recorders to achieve echo and delay voice transformations. Voice effects in music production became really popular in the mid sixties and seventies basically due to the advances in electronics and also due to the experimentation spirit of those days. Actually, it is really rare (nearly impossible) to find a popular music production in which no effect has been applied. Furthermore, it is very usual to apply alien modifications to the voice to transform it up to the point the voice loses its human character (especially in more dance and electronic productions).

In this sub-workpackage we offer voice transformations which, using state of the art signal processing algorithms, manage to convert and modify the voice of the users in amazing, yet musically meaningful ways that preserve the natural qualities of the voice.

Voice transformation is achieved with the *ComboVox* plugin, which allows the user to sing along with the music played by the system, while having his or her voice corrected or modified in meaningful transformations, such as adding vibrato, transposing or correcting pitch or modifying timbre. With *ComboVox* users are able to transform the character of their voice in real-time. It allows transformations such as *gender change*, *robot effect*, *ogre effect*, and others. It is based on manipulations of pitch and timbre.

2.1.2 Method description

The underlying technique in the voice conversion process is called Voice Pulse Modeling. This approach tries to combine the strengths of classical time and frequency domain techniques into a single framework, providing both an independent control of each voice pulse and flexible timbre and phase modification capabilities. A more complete and detailed description of the algorithm can be found [Bonada, DAFX 04]. Figure 1 schematizes this technique.

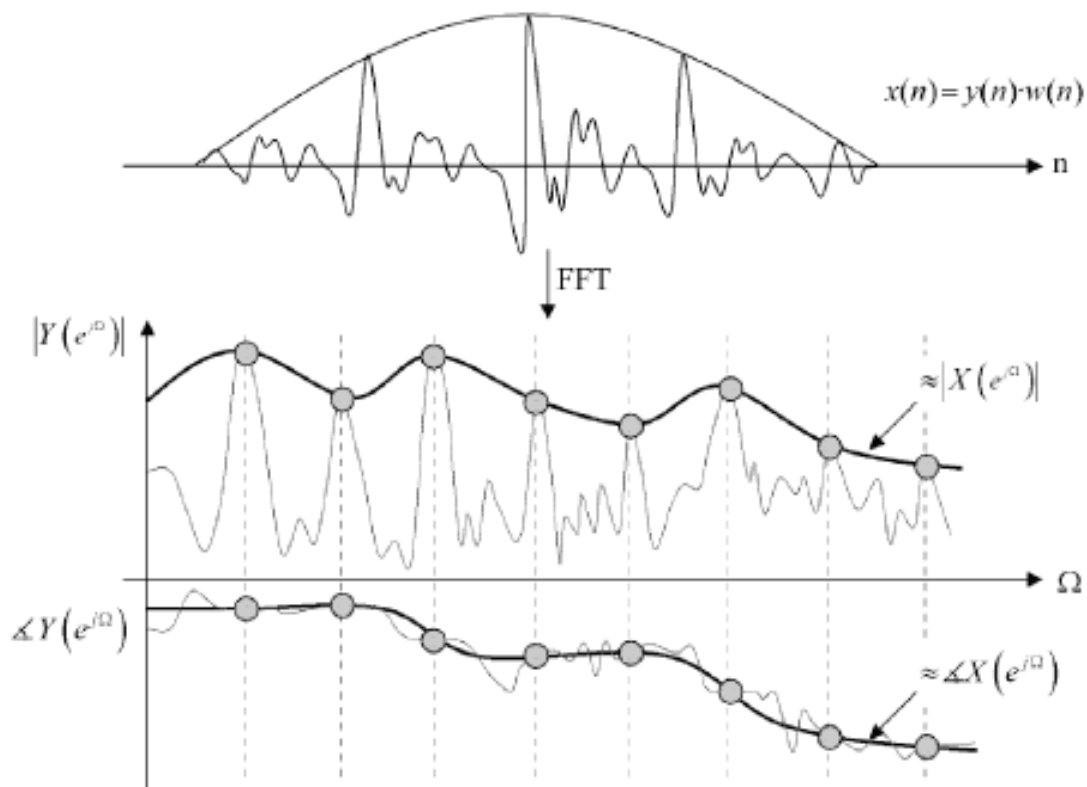


Figure 1: Spectrum estimation of a single radiated voice pulse

2.1.3 Position over state-of-the-art

Singing Voice Processing has been addressed mainly with Analysis/Synthesis approaches. At IRCAM, researchers developed for the film “Farinelli” a tool that imitates the voice of a castrato singer. A synthesized voice was created by recording a coloratura soprano and a counter tenor, and by applying morphing techniques to generate the new voice with the characteristics of a castrato [Depalle04]. Most of the research being carried in the last decade is based on phase-vocoder techniques [e.g. Puckette95, Abe90, Laroche99]. On the other hand, today we can find several commercial systems targeted to professional studios that apply voice transformations. In terms of quality, the system developed by the company TC-Helicon stands out [Helicon].

In our labs, we started working on voice transformation in 1997 in a project we called Elvis [Bonada01][Cano00]. The goal of the project was to achieve real time voice conversion (impersonation) for the Japanese *karaoke*s. For that project we developed a Linux-based prototype for real time transformations based on the Spectral Modelling Synthesis (SMS) sinusoidal plus residual decomposition of the sound [Serra89].

Recently we have been working on moving the voice transformations to phase-locked *vocoder* like techniques. In this attempt, we have also tried to achieve natural transformations specifically related to voice, such as applying or modifying the vibrato or the roughness, standing on an excitation / vocal tract production model of the voice called *Excitation plus Residual* (EpR), as shown in Figures 2 and 3.

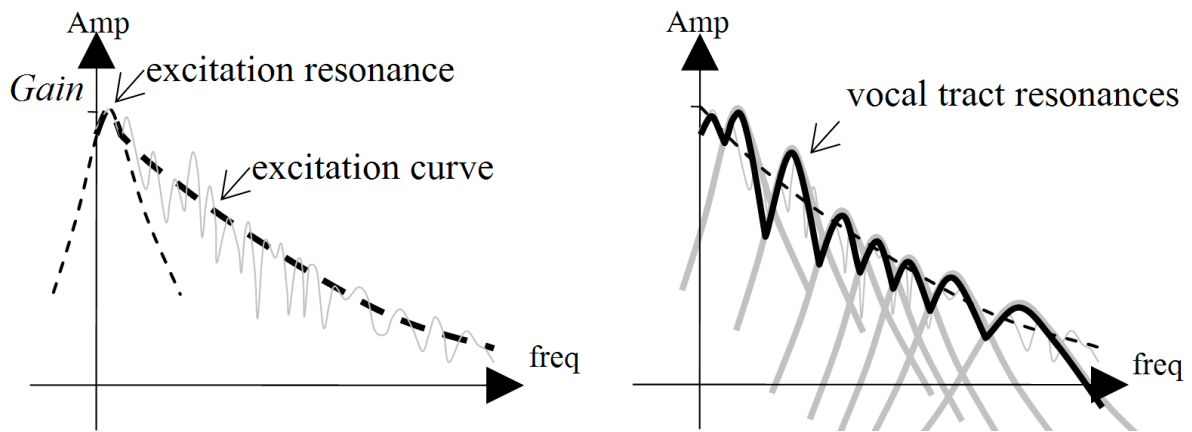


Figure 2: The EpR model

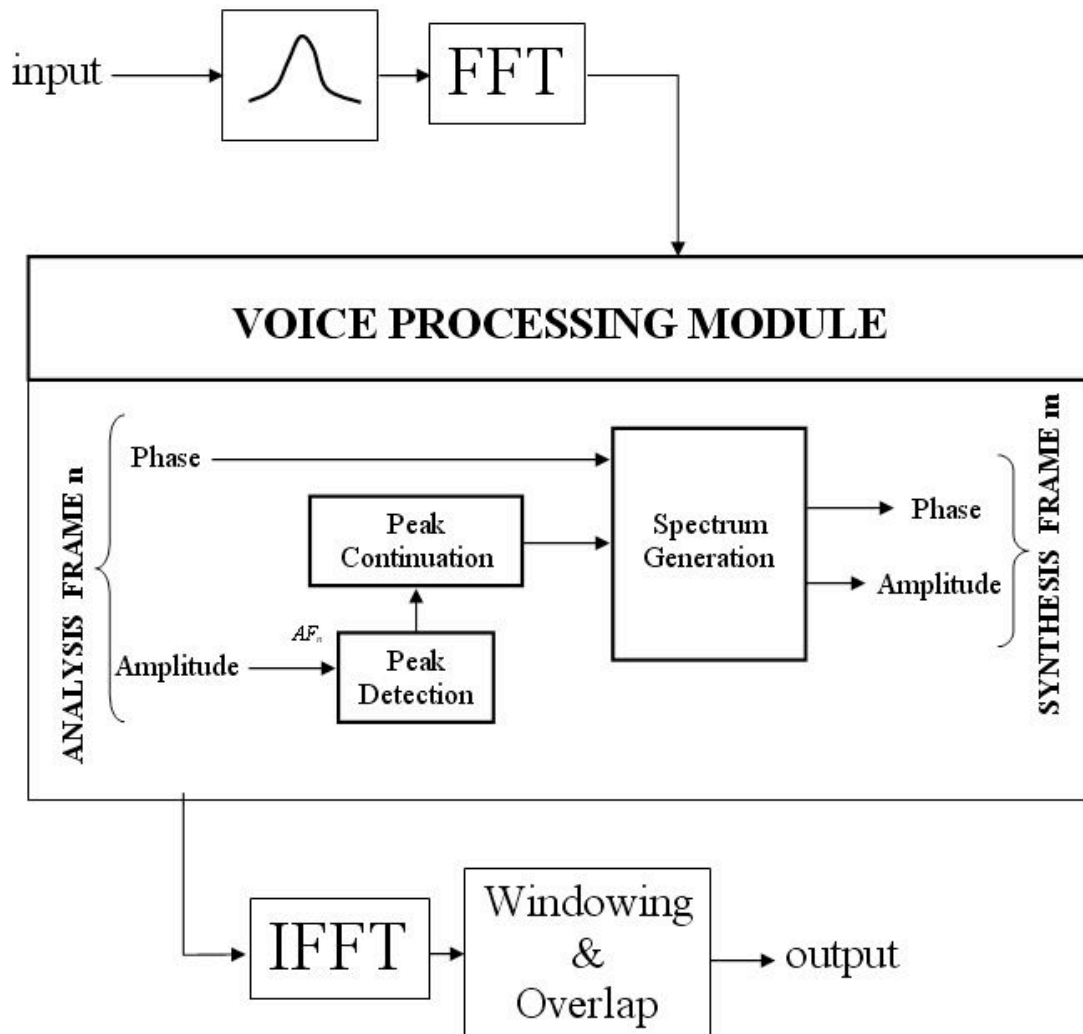


Figure 3: The EpR mechanism

This processing is achieved in the spectral domain. The previous figures give a more detailed description of the approach. Given a spectral frame n , which contains amplitude and phase of the FFT bins, a series of spectral peaks is found. Then, voice transformations are applied directly to the spectral peaks information. Finally, a rendering module generates the spectrum (amplitude and phase) of the output synthesis frame. By means of the IFFT and a windowing & overlap mechanism, the signal is transformed back to the time domain.

The aforementioned mechanism allows for radical and yet, still natural transformations of the voice. These transformations include “classical” effects, such as state-of-the art pitch-shifting which preserves the timbre characteristics of the input voice, and at the same time, also allows for less “typical” and more radical transformations, such as modifying the genre, the whisper or the growl of the input voice, but still preserving the voice naturalness.

The major breakthrough of this sub-workpackage consists in porting a set of transformations which traditionally were almost exclusive to time-domain techniques, such as roughness, breathiness or formant shift, to the spectral domain techniques.

- [Abe90] Abe, M., Nakamura, S., Shikano, K. & Kuwabara, H. (1990). Voice Conversion Through Vector Quantization, *J.Acoustical Society, Japan*, (E) 11(2), 71-76.
- [Bonada01] Bonada, J. Celma, O. Loscos, A. Ortolà, J. & Serra, X. (2001). Singing Voice Synthesis Combining Excitation plus Resonance and Sinusoidal plus Residual Models. In *Proceedings of the International Computer Music Conference 2001*
- [Cano00] Cano, P., Loscos, A., Bonada, J., de Boer, M. & Serra, X. (2000). Singing Voice Impersonator Application for PC. In *Proceedings of the International Computer Music Conference*.
- [Depalle04] Depalle, P., García, G. & Rodet, X (1994). A virtual Castrato. In *Proceedings of the International Computer Music Conference, Aarhus, Denmark, 1994*.
- [Helicon] TC-Helicon, Voice Prism: <http://www.tc-helicon.com/>
- [Laroche99] Laroche, J. & Dolson, M. (1999). 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*.
- [Puckette95] Puckette, M. S. (1995). Phase-locked vocoder. *Proc. of IEEE Conference on Applications of Signal Processing to Audio and Acoustics*, Mohonk.
- [Serra89] Serra, X. (1989). A System for Sound Analysis/Transformation/Synthesis based on a Deterministic plus Stochastic Decomposition. *Ph.D.Dissertation, Stanford University*.

2.1.4 **Benchmarks**

The modules integrated in the *Performance* workpackage (WP5) must satisfy the condition of working in real-time. On the other hand, latency plays an important role in any interactive system. Our proposed system requirements are based on the platforms we used for testing. In the following tests we obtained a maximum latency of 30 ms, which although not optimal is still musically useful and very reasonable considering today's standards and technologies.

Latency	$\leq 30\text{ ms}$
---------	---------------------

In order to have a first evaluation of the developments in the WP5, which were not tested within the Authoring Application, we carried out user tests at the UPF facilities during the last week of July 2006. Although the test dates were later than initially expected, we decided to carry out the experiments during the S2S2 Summer School in Sound and Music Computing (<http://www.iaa.upf.es/mtg/sssmc2006/>), which took place on the same dates.

Technical setup

- Computer
 - PC Pentium 4, 2.4 GHz 1 GB RAM
 - Windows XP Professional
- Audio
 - Edirol UA-25 USB audio interface
 - Loudspeakers Edirol MA-10A
 - Microphone Shure SM-58

For running the tests, we used an evaluation version of the AudioMulch audio software (www.audiomulch.com). This software is very indicated for this type of tests with VST plugins due to its flexibility and low computational and memory requirements.

Task

The task is to perform karaoke songs with a transformed voice to increase the engagement. Two karaoke songs (with only the instrumental parts) were available: “Yesterday” and “Bésame mucho”. Users were asked first, to test all transformations and, second to perform a song with their preferred transformation.

Feedback

All users commented they could imagine a HiFi system with karaoke features, in which voice transformations could be integrated. Some users, in addition pointed out that the “fiction” transformations would be not interesting in such a system.

2.1.5 Implementation

The *ComboVox* is the current implementation of the voice transformation techniques described above. It is a VST-plugin for boosting voice in karaoke performances. The plugin offers singing meaningful transformations as vibrato or timbre modification. It has been developed in C++ and runs in real-time under any compatible VST hosts. Following is an overview of each of the transformations that can be achieved with the *ComboVox*.

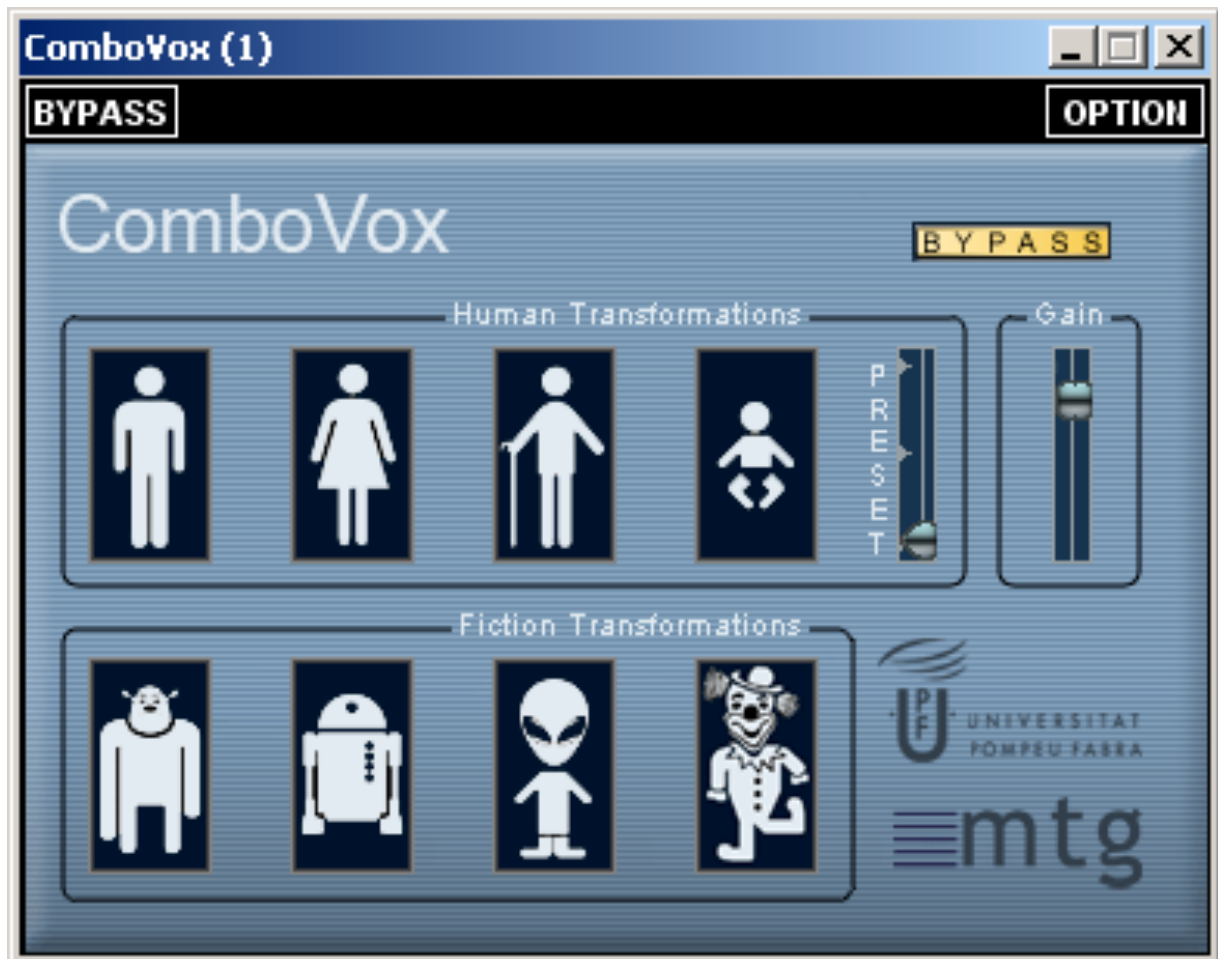


Figure 4: The graphical user interface of the ComboVox plugin

User Controls

The principal objective of this plugin is to offer voice transformations to a wider audience, and thus, a simple use interaction is preferred. As seen in the graphical user interface shown in Fig. 4, a set of eight buttons cover all possible transformations, grouped in *Human Transformations* and *Fiction Transformations*. In the following table, we describe the characteristics of each transformation.

**HUMAN
TRANSFORMATIONS**

Male: gives the voice a distinct male character

Female: gives the voice a distinct female character

Older: ages the voice to make it sound like an old person

Child: rejuvenates the voice to make it sound like a child

**FICTION
TRANSFORMATIONS**

Ogre: makes the voice resemble a horrible giant beast's voice

Robot: robotizes the voice and makes it sound electronic and metallic

Alien: gives the voice a weird out of space personality

Comic: makes the voice sound like a cartoon character

CONTROLS

Gain: controls the output gain

Preset Selector: switches among the three different presets for each human transformation

2.1.6 Dissemination materials**2.1.6.1 *Scientific publications***

- [1]. Fabig, L. and Janer, J., "Transforming Singing Voice Expression - The Sweetness Effect", *Proceedings of 7th International Conference on Digital Audio Effects*, 2004.
- [2]. Janer, J. Bonada, J. Blaauw, M.; "Performance-driven control for sample-based singing voice synthesis"; *Proceedings of 9th Intl. Conference on Digital Audio Effects*; Montréal, 2006
Other scientific dissemination actions.
- [3]. Jordà, S., Kaltenbrunner, M., Geiger, G., Bencina, R. "The reacTable*", *Proceedings of International Computer Music Conference, Barcelona*, 2005.
- [4]. Loscos, A. and Bonada, J., "Emulating Rough And Growl Voice In Spectral Domain", *Proceedings of 7th International Conference on Digital Audio Effects*, 2004.

2.1.6.2 *Public presentations*

Alox Loscos, "The Semantic HiFi Project. Performance prototypes". School of Computing, National University of Singapore, 15/5/2006.

2.1.6.3 *Products*

ComboVox. Add-on plugin for the Pinnacle Studio 10 Bonus DVD.

<http://www.pinnaclesys.com/>

2.1.6.4 Related PhDs

Loscos, A. *Singing Voice Transformations on Spectral Domains*, UPF Ph.D. Dissertation, 2006.
(Thesis defense requested)

2.2 WP5.2 Voice Instrumentizer

Responsible partner : UPF

2.2.1 Functional description

Voice instrumentation is a step further into voice transformation. In this case, the voice of the performer is not used anymore as a modified voice; instead the maximum information from the voice is extracted and mapped into different instrumental synthesis algorithms, so to allow singers to directly control different musical instruments such as bass, drums or synthesizers that will be meaningfully added to the music playback.

This sub-workpackage seeks to explore the possibilities of the singing voice as a natural and simple, yet sophisticated musical instrument. It provides new ways of control for digital musical instruments, accomplishing at the same time an expression control improvement for musicians and a low-entry cost for novices.

2.2.2 Method description

A voice analysis module extracts information about the characteristics of the voice signal in real-time. Depending on these descriptors, the synthesis module takes one sample from a database and transforms it. The attained effect is that the voice drives an instrument sound, or what we call “instrumentizing” the voice.

In the current implementation two instrument sounds are integrated: bass guitar and flute. With the taken instruments, we cover different instrument families which have usually distinct musical purposes. Bass guitar sounds are mainly used as rhythm accompaniment, while the flute acts usually as soloist.

In the GUI, one slider allows transposition of one octave up/down. Another slider allows to change the envelope amplitude.

2.2.3 Position over state-of-the-art

Sound and Music Computing research field has tackled the singing voice mainly from the analysis / synthesis perspective, putting efforts in “inventing” a computer that sings as a human. Starting in the sixties but still a hot topic, the research experimented an important progression. Recently, an approach of singing voice synthesizer based on sample concatenation has resulted in a commercial product [Bonada03].

Controlling sound synthesis with another audio stream has been addressed in the past. A system developed by the Hyperinstruments Group at the Massachusetts Institute of Technology, under the direction of Tod Machover, shares the same objectives of the

Performing Voice Instrumentizer. This system was developed by Tristan Jehan, employing previous work done by Bernd Schoner and Eric Metois. Finally, the system's description was compiled in [Jehan01]. In another approach by Miller Puckette et al. [Puckette04] 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. Another recent related project is documented in [Pöpel05].

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 [Puckette95].

With the approach developed for the Semantic-HiFi project, we introduce new ways of controlling digital musical instruments. We consider two different stages, Voice Analysis and Instrument Synthesis.

- [Bonada03] Bonada, J. and Loscos, A. (2003). Sample-based singing voice synthesizer by spectral concatenation. In *Proceedings of Stockholm Music Acoustics Conference*, 2003.
- [Puckette95] Puckette, M.S. (1995). Score following using the sung voice. In *Proceedings, International Computer Music Conference*, 1995.
- [Cano99] Cano, P., Loscos, A. and Bonada, J. (1999). Score-Performance Matching using HMMs. In *Proceedings of the International Computer Music Conference*, 1999.
- [Jehan01] Jehan, T. (2001). *Perceptual Synthesis Engine: An Audio-Driven Timbre Generator*. Master Thesis.
- [Puckette04] Puckette, M. (2004). Low-dimensional parameter mapping using spectral envelopes. In *Proceedings of the International Computer Music Conference*, 2004.
- [Pöpel05] Pöpel, C. and Dannenberg, R. (2005) Audio signal driven sound synthesis. In *Proceedings of the International Computer Music Conference*, 2005.

2.2.4 **Benchmarks**

As already stated, the modules integrated in the *Performance* workpackage (WP5) must satisfy the condition of working in real-time. On the other hand, latency plays an important role in any interactive system. Our proposed system requirements are based on the platforms we used for testing:

In order to have a first evaluation of the developments in the WP5 that were not tested within the Authoring Application, we carried out user tests at the UPF facilities during the last week of July 2006. Although the test dates were later than initially expected, we decided to carry out the experiments during the S2S2 Summer School in Sound and Music Computing (<http://www.iaa.upf.es/mtg/sssmc2006/>), which took place on the same dates.

Technical setup

- Computer
 - PC Pentium 4, 2.4 GHz 1 GB RAM
 - Windows XP Professional
- Audio
 - Edirol UA-25 USB audio interface

- Loudspeakers Edirol MA-10A
- Microphone Shure SM-58

Task

For the **Voice Instrumentizer**, the task to accomplish was to add a bass line to a jazz trio (sax, piano and drums) recording, in which the bass was removed. In this experiment, users could practice without the music recording to learn how to control the experiment.

Feedback

Most users answered it was interesting and useful at some points. In the comments, some people added however that they did not see a clear situation where to use it. Although, most of the users answered it was interesting in the first question, in general they found it as “not adequate” for HiFi systems.

Users that were musicians found it as “somehow adequate” or as “very adequate” in a DJ context. In the comments, they pointed out it could help the engagement of DJ sessions, increasing the interaction with the audience.

2.2.5 Implementation

In the *Voice Instrumentizer* plugin, a monophonic instrument sound is controlled by means of the user’s voice. This plugin allows adding a musical line on top of a selected musical piece.

Essentially, we aim to extract vocal gestures that can be of three different kinds in the voice production mechanism, depending on whether its origin is either in the breathing system, in the glottis, or in the vocal tract. In this approach, the chosen features are classified based on control aspects: Excitation, Vocal Tract and Context.

For the synthesis stage, our Spectral Model approach combines a sample-based synthesis with transformations, based in the Spectral Peak Processing. A particularity of our sample-based algorithm is that it works in the frequency domain, as shown in Figure 5. 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 periodically the spectral frames and transform some characteristics using the SPP framework. Finally, the processed spectral frames are converted to the time domain through the inverse Fourier transform.

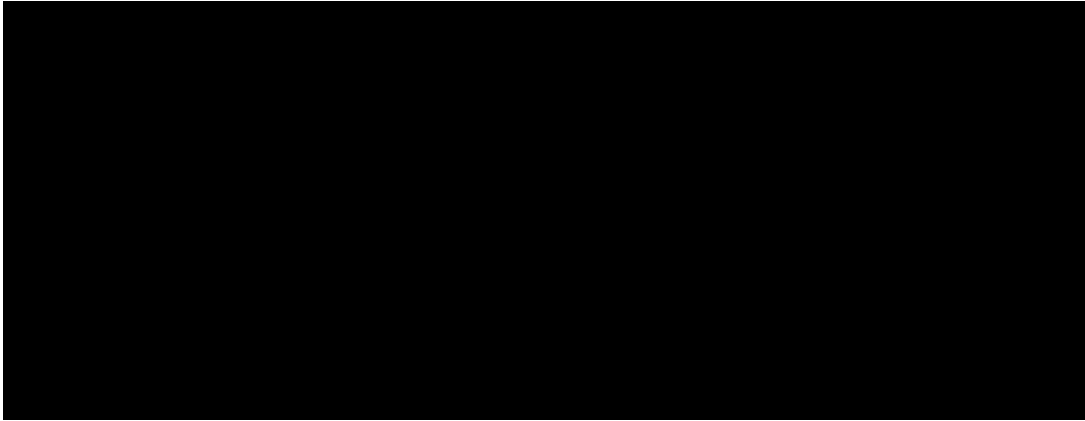


Figure 5. System's overview.

For our bass synthesizer implementation, we set up a small database consisting of 12 template tracks (containing one note sample). The tracks are analyzed off-line in the spectral domain, and stored in the database in the form of binary files containing spectral data (complex spectrum, harmonic peaks, estimated pitch, etc.). In a further step, all tracks were labelled by hand according to its characteristics. Currently, three features were annotated: *Pitch*, *Dynamics* and *Attack Type*. The pitch values are specified in Hz, while *Dynamics* and *Attack Type* are normalized (range is $[0..1]$). In the case of *Dynamics*, 0 corresponds to a *pp* sound, and 1 to a *ff*. The attack type is a concept, which will depend on the instrument synthesized. Concerning bass sounds, we have decided to classify two types of sounds, depending on the plucking technique which can be slap or fingered, with resulting differences primarily related to the attack.

Our last improvements focused on timing issues of the synthesis process. The Sample database contains a number of analyzed samples, which have a specific duration. In a performance situation the duration of the note is unknown, thus we need a strategy to modify the length of the note. This strategy is basically *looping*. Depending on the characteristics of the instrument, we will apply different looping methods. For plucked string instruments such as the bass guitar, the technique that is applied consists in repeating a selected spectral frame, while continuing the phase in a phase-locked vocoder synthesis. Figure 6 shows the waveform of voice, original bass guitar sample, and the morphed version.

For the flute sound, the sustain region is specified, and the sample is looped as long as the current voice gesture is active.

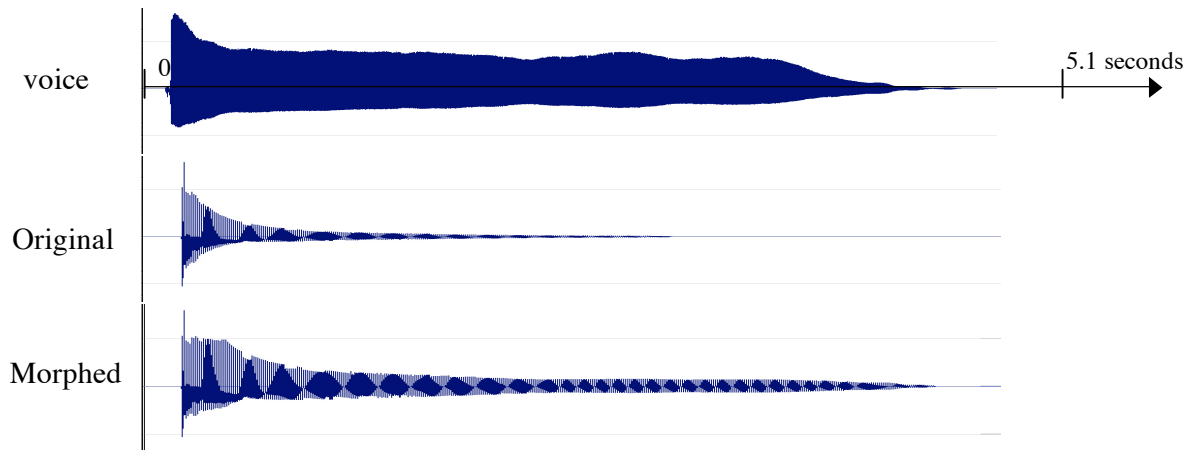


Figure 6. Voice waveform, original bass guitar sample, morphed sound

As depicted in Figure 7, the GUI is kept simple allowing instrument selection and a few controls for transpose, Sensitivity and Envelope. In the next table user controls are described in detail.

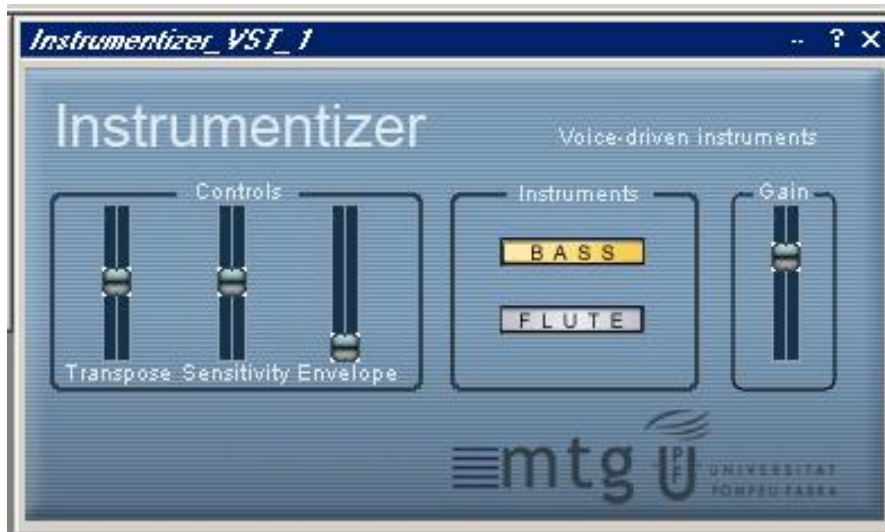


Figure 7. The Instrumentizer GUI

User Controls:

Controls	
TRANSCOPE	It transposes the output sound in a range of one octave up/down. Note that depending on the instrument to synthesize, this slider has to be adjusted in order to achieve a more natural sound.
SENSITIVITY	This control is used to adjust the input signal level since it may vary depending on the microphone, pre-amplifier, etc. It mainly affects the note triggering.
ENVELOPE	Varying this slider, we are able to create new and unheard sounds such as a long sustained bass guitar note. Placing the

	slider in its top position, the amplitude envelope of the output sound is taken from the input voice. Putting the slider in its bottom position, the amplitude envelope of the output sound is taken directly from the sample in the database. In this case, it may occur that the output sound fades out while having still energy in the input voice
<i>Instrument:</i>	
BASS	It selects a bass guitar as target instrument. The synthesized bass guitar sound is based on samples. Currently, only samples of a single bass guitar are provided.
FLUTE	It selects a flute as target instrument. The synthesized flute sound is based on samples. Currently, only samples of a single flute are provided.

2.2.6 Dissemination materials

2.2.6.1 *Scientific publications*

- [1]. Janer, J. (2005). Voice-controlled plucked bass guitar through two synthesis techniques. In *New Interfaces for Musical Expression Proceedings*, Vancouver, Canada, 2005.
- [2]. Janer, J., “Feature Extraction for Voice-driven Synthesis”, *Proceedings of the AES 118th Convention*, 2005.
- [3]. Janer, J., Loscos, A. “Morphing techniques for enhanced scat singing”, *Proceedings of 8th International Conference on Digital Audio Effects*, 2005.
- [4]. Loscos, A. and Celma, O., “Larynxophone: Using Voice As A Wind Controller”, *Proceedings of International Computer Music Conference*, 2005.

2.2.6.2 *Public presentations*

Presentation. Jordi Janer. “Voice as a musical controller”. McGill University, Music Technology Area, Montréal, Canada. 15/10/2005.

2.2.6.3 *Related PhDs*

Janer, J. *Voice Control for Digital Musical Instruments*, UPF Ph.D. Dissertation, 2006. (*Thesis defense requested*)

2.3 WP5.3 DJ Voice-Controller

Responsible partner : UPF

2.3.1 Functional description

The **Wahwactor** is a two-input and one-output system. Out of the two input tracks, one of the tracks may be considered a control rather than a proper input since it is the one in charge of driving the transformations to be applied to the audio signal. In the context of the Wahwactor, the audio signal is typically a guitar signal and the control signal is a voice [wa-wa] utterance signal.

First, the voice signal is analyzed to pick up a meaningful descriptor that, after a simple conversion (shift, scale and smooth), is used as the centre frequency of the wah-wah filter, through which the guitar signal is sent to be mixed with the original.

2.3.2 Method description

To work in real-time, the Wahwactor uses a frame-by-frame algorithm described by the diagram illustrated in Figure 8. The voice signal is sampled at 44100 Hz and analyzed using a 2100 sample Hamming window and a 2048 point Fast Fourier Transform. The guitar signal is sampled at 44100 Hz and filtered using 2100 sample length buffers. The algorithm uses a 1050 sample hop size so that we have a 50% overlap in synthesis. This overlap is necessary to smooth the filter phase frame to frame variations.

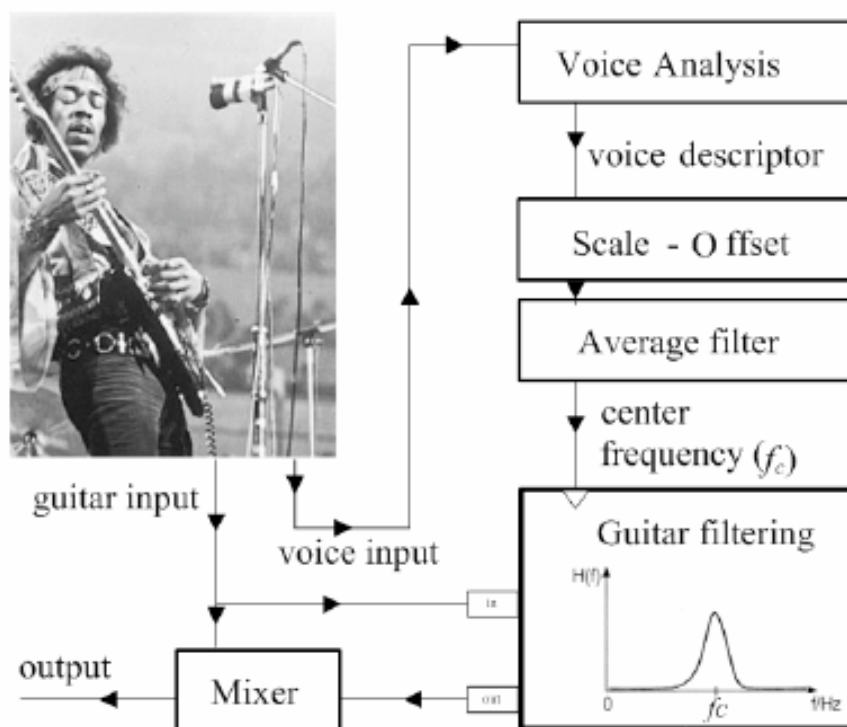
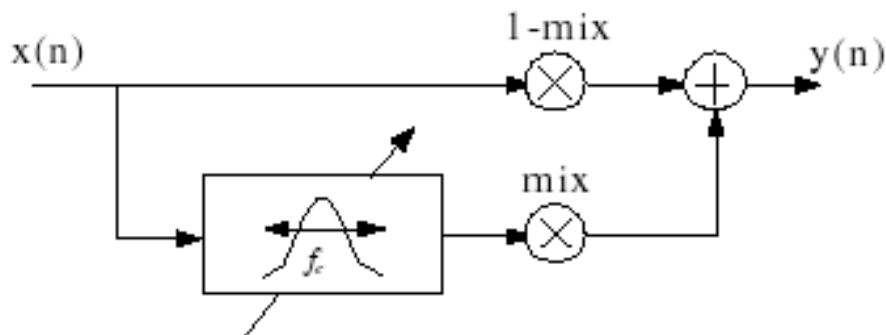


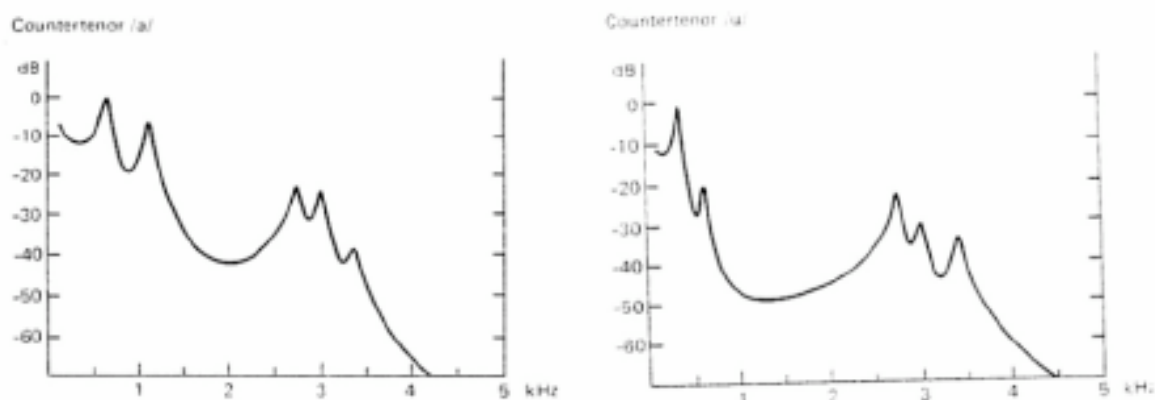
Figure 8: the Wahwactor block diagram.**2.3.3 Position over state-of-the-art**

Although the wah-wah effect was initially developed by trumpet players using mutes in the early days of jazz, it has become known as a guitar effect ever since Jimi Hendrix popularized Vox Cry-baby pedal in the late 60's.

A wah-wah guitar pedal contains a resonant band pass filter with a variable center frequency that is changed by moving the pedal back and forth with your foot. Usually, a knob controls the mix between original and filtered guitar signals as represented in Figure 9.

**Figure 9: general wah-wah effect block diagram.**

The explanation of why the wah-wah effect resembles human [wa-wa] utterance is found on the voice spectral characteristics of the vocalic phonemes [u] and [a], in particular, on the first formant location. Considering the [u] vowel first formant is around 350 Hz, and the [a] vowel first formant is around 700 Hz, the [u] to [a] articulation produces a modulated sound due to the trajectory of the first formant that is perceived as the effect of a resonant filter moving upwards in frequency. This is shown in Fig. 10.

**Figure 10: Spectral envelopes of vowels [a] (left) and [u] (right), the formants different locations are clearly distinguishable.**

One of the initial attempts to control audio effect with the mouth cavity can be credited to the *TalkBox*. It was a device consisting of tube driving the sound to the mouth and a microphone

that captured it. The result was a sound shaped with the mouth articulation, and was mainly employed to have “talking guitar”. It was popularized by Peter Frampton in 1973.

More recently, a musical interface prototype developed by ATR Media Integration & Communication Research Labs profits from the link between the wah-wah effect and the [wa-wa] utterance. The system, called *Mouthesizer* [Lyons01], uses a video camera to measure the opening of the performer's mouth and changes the wah-wah filter centre frequency according to this measure. It was in fact the multifaceted requirements of such a system what made us think about an alternative straightforward solution.

[Lyons01] Michael J. Lyons, Nobuji Tetsutani., "Facing the Music: A Facial Action Controlled Musical Interface", Proceedings CHI 2001, Conference on Human Factors in Computing Systems, March 31 - April 5, Seattle, pp. 309-310.

2.3.4 Benchmarks

The latency is approximately 10 ms for sound I/O using ASIO drivers and 24 ms due to the analysis hop size.

Latency	34 ms
---------	-------

In order to have a first evaluation of the developments in the WP5 that were not tested within the Authoring Application, we carried out user tests at the UPF facilities during the last week of July 2006. Although the test dates were later than initially expected, we decided to carry out the experiments during the S2S2 Summer School in Sound and Music Computing (<http://www.iaa.upf.es/mtg/sssmc2006/>), which took place on the same dates.

Technical setup

- Computer
 - PC Pentium 4, 2.4 GHz 1 GB RAM
 - Windows XP Professional
- Audio
 - Edirol UA-25 USB audio interface
 - Loudspeakers Edirol MA-10A
 - Microphone Shure SM-58

Task

The task in this case was to modify the sound result of a mix with voice controlled FX. This mix consisted of two separate tracks: electric guitar and bass+drums. The user applied the Wah-Wah effect to the guitar solo track, and the transformed sound was mixed with the drum+bass track. Users could practice with the system before starting with the actual test.

Feedback

Users stated that in a HI-FI system, this tool would be either "completely adequate" or "not very adequate". About the integration as a DJ-Tool, opinions were quite different. Users who considered that it was "very adequate" explained that controlling effects with the voice is a very nice tool for DJs, especially since very often they already use their hands while mixing. On the contrary, users considering that it was "completely inadequate" explained that the wahwactor is a nice tool but senseless for applications different from the guitar playing.

2.3.5 Implementation

The *Wahwactor* is a VST-plugin with which the user can apply the wahwah effect over an instrument controlling its modulation by using [wawa] voice utterances. The plug-in also extends the voice control to phaser, flanger, chorus and tremolo effects. It has been developed in C++ and runs in real-time under any compatible VST hosts. All transformation algorithms are based on spectral techniques that analyze and modify the content in frequency domain. Let's have an overview of each of the transformations we can achieve with the *Wahwactor*, shown in Figure 11.



Fig. 11. The graphical user interface of the Whawactor plugin

- Wahwah: when button is on, the slider controls the bandwidth of a peak filter. The frequency of this filter is modulated according to the voice utterance.
- Phaser: when button is on, the slider controls the bandwidth of a notch filter. The frequency of this filter is modulated according to the voice utterance.
- Flanger: when button is on, the slider controls the delay offset to be applied in a flanger effect system. Over the offset, the modulation of the delay is applied according to an envelope computed out of the voice utterance.
- Chorus: when button is on, the slider controls the delay offset to be applied in a chorus effect system. Over the offset, the modulation of the delay is applied according to an envelope computed out of the voice utterance.
- Tremolo: when button is on, the amplitude of the signal can be modulated by a sinusoid like envelope computed out of the voice utterance (sine option) or by a quantification of this envelope that turns it into a square like envelope (square option).
- Mix: the slider controls the dryness / wetness of the effect.
- Volume: the slider controls the gain of the effect.

2.3.6 Dissemination materials

2.3.6.1 *Scientific publications*

- [1]. Loscos, A., Aussenac, T., "The Wahwactor: a voice-controlled pedal", Proceedings NIME 05, International Conference on New Interfaces for Musical Expression, May 26-28 2005, Vancouver, Canada.

2.3.6.2 *Other scientific dissemination actions*

<http://www.iaa.upf.es/~taussenac/wahwactor.htm>

2.4 WP5.4 Rhythm transformation

Responsible partner : UPF

2.4.1 Functional description

This feature was initially intended to allow, “conducting or modifying the tempo and/or the dynamics of the music being played by the system using the HIFI remote command as a baton or tapping the beat”. Since during the evolution of the project, the consortium decided that no additional sensors would be implemented in the remote command, the feature has been finally implemented, allowing the user to tap the beat in some button or key.

This feature involves three complementary techniques: (1) detection of the user periodic inputs, (2) on-the-fly detection of the background music beats, so that tempo changes can be synchronized with the user, and (3) high-quality time-stretching and pitch preserving techniques, which will allow the continuous modification of the music playback speed, without affecting its pitch nor its timbre. These modifications also include the possibility of changing the swing feel of the song (without affecting the average tempo), either by conducting or, simpler, by means of a dedicated “swing” slider.

Figure 12 shows the different mechanisms to control the Groovator in a performance situation: Graphical User Interface, MIDI controller and Input audio stream.

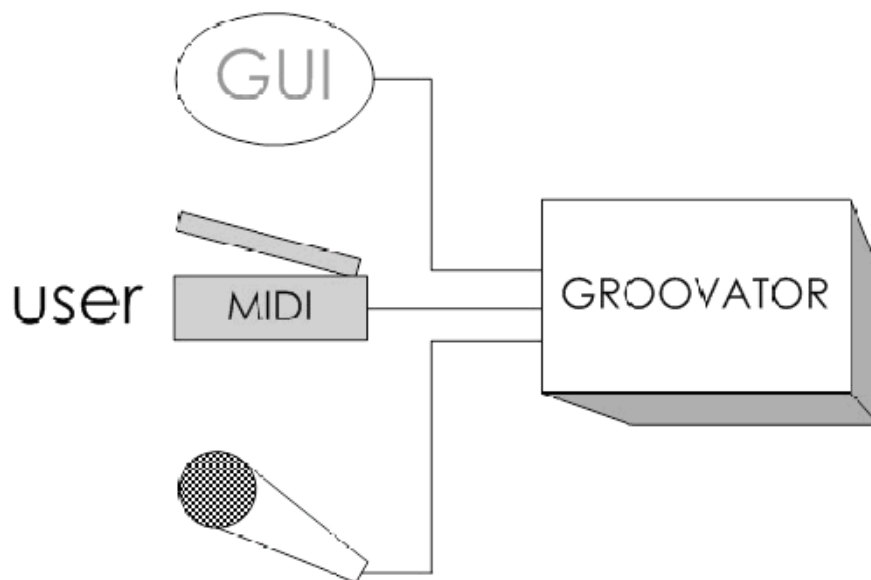


Fig. 12. The Groovator scheme

2.4.2 Method description

A rhythm analysis module extracts information of tempo and beat location. Based on this rhythm information, we apply different transformations: "Tempo variation", and "Groove". This type of manipulation is generally referred as Content-based transformations. In addition, user interaction plays also an important role in this system. Tempo variations can be controlled either by tapping the rhythm with a MIDI interface or by using an external audio signal as tempo control. We foresee several use-cases, focusing on live performance situations.

2.4.3 Position over state-of-the-art

Our main requirements are high quality, flexible time-scaling ratios and low computational cost. From this perspective, time-domain algorithms are not a good choice because they give best results for small modification factors with single signal sources, even in noisy situations and fail to deal with polyphonic material. Phase-vocoder techniques give smoother results for large time-scale factors and work well with polyphonic material. However they introduce smearing for impulsive signals and smooth transients. On the other hand, recent improvements on these techniques have successfully minimized the reverberance or phasiness introduced by the loss of vertical phase coherence, and some multiresolution approaches on top of the phase-vocoder perform spectral analysis closer to that performed by the human auditory system. On the other hand, signal models have shown their ability to split the input signal into different components which can be processed independently. Basically these components are sinusoids, transients and noise. This decomposition gives a lot of flexibility when thinking on transformations. Sinusoids are good to model quasi-stationary components (slow-varying partials), and can deal with impulsive signals with a waveform preservation technique based on phase delays, thus preserving the synchronization of the harmonics' phase, but only in the case of single signal sources. Several advances have been achieved in estimating the sinusoid parameters, including multiresolution approaches, which results in higher sound quality and less pre-echo, but there is much work yet to do. Transients are good to model attacks and fast changes. In fact, time-scale of transients means nothing more than translating them into a new onset position. They are detected and parameterized from the residual obtained by subtracting the sinusoids from the original sound. The noisy residual is obtained from this first residual by subtracting the transients. The noise is modelled as slowly varying filtered white noise, which is a good model for time-scaling breath or air flow of instruments for example. We can say that these signal models are very powerful and give much more flexibility than time-domain or phase-vocoder techniques. Very extreme time-stretch ratios can be applied producing high quality results; even the signal can be slowed down to a frozen state. However, with no transformations, they don't allow a perfect reconstruction of the original signal, unless the noisy residual is not modelled but just kept as it is. Thus, the synthesized output signal does not sound exactly as the original one, but very close. This is a drawback for professional post-production applications.

In our context it seems that phase-vocoder based techniques are the best option. They allow a perfect reconstruction when no transformation is applied and work well with polyphonic sources. Our time-scaling technique is based on these techniques. However, several enhancements are included, especially to deal with transients and impulsive signals in a real-time environment using stereo files. First of all, we use a constant hop size in our processing so that small audio frames are repeated or dropped depending on the time-scale ratio. This allows going beyond the time-scale ratio limit typical of the phase-vocoder (because a

minimal window overlap is required in synthesis). Secondly, transients are detected and processed in a way that their perceptual attributes remain quite unmodified after the time-scale transformation. Besides, a multiband approach is used in order to emulate the frequency response of the human auditory system and improve the overall sound quality. In addition, multirate processing is used for improving the computational cost of our multiband approach. Finally, an efficient technique for preserving the aural image of stereo files is also used in our system, avoiding the typical problem of audio objects flying through the aural image.

Regarding explicitly *Rhythm transformations*, in the literature, the type of transformations addressed here are referred as Content-based transformations or Adaptive Digital audio Effects. Basically, it means that the transformation depends on the input signal analysis.

Several approaches proposed innovative rhythm manipulations, either altering prerecorded material [3] or as extensions of the Beat-Boxing retrieval [5, 6]. A rhythm analysis module extracts tempo and time location of every beat in the beat sequence. In the areas of Music Cognition and Music Computing, *rhythm induction* has widely attracted the interest of researchers, describing different rhythmic representations and algorithms [2, 4].

- [1] Bonada, J., *Automatic Technique in Frequency Domain for Near-Lossless Time-Scale Modification of Audio*, Proceedings of the International Computer Music Conference, Berlin, Germany, 2000.
- [2] F. Gouyon. A computational approach to rhythm description — Audio features for the computation of rhythm periodicity functions and their use in tempo induction and music content processing. PhD thesis, Universitat Pompeu Fabra, Barcelona, 2005.
- [3] F. Gouyon, L. Fabig, and J. Bonada. Rhythmic expressiveness transformations of audio recordings: swing modifications. In Proc. Int. Conf. on Digital Audio Effects (DAFx-03), London, pages 94–99, 2003.
- [4] H. Honing. Structure and interpretation of rhythm and timing. *Dutch Journal of Music Theory*, 7(3):227–232, 2002.
- [5] G. Tzanetakis, A. Kapur, and M. Benning. Query-by-beat-boxing: Music retrieval for the dj. In ISMIR-2004, 2004.
- [6] O. Gillet and G. Richard. Drum loops retrieval from spoken queries. *Journal of Intelligent Information Systems*, 24:2/3:159–177, 2005.

2.4.4 **Benchmarks**

The current implementation runs on any VST host application. In terms of computational load, the system uses 14% and 22% of CPU time, for the playback and streaming implementations respectively. Measurements were completed on a PC Pentium IV, 2.4GHz running Windows XP. The audio interface latency was set to 512 samples using ASIO drivers.

Also, in the case of the *Groovator*, user tests were carried out by Native Instruments personnel in Berlin. The *Groovator* tests were actually executed within the Authoring Application test sessions, using the *Groovator* as a VST plugin in the Authoring Tool.

Task

In this case, the task to accomplish was to apply rhythm transformations to an existing audio file. The user select one audio file, and applies either tempo variation or swing (groove) variation.

Feedback

After a short explanation of the effect at the beginning of the test most test persons expressed already that it was not clear, how to use such an effect. Even many of the testable functions remained incomprehensible ("Inertia", "Look ahead", "Calibr", "Auto"). Only the "Factor" function was understood by most users. Partly it was expressed that this effect could be used for short passages; however the majority of the test persons regarded this effect as not suitable for a live performance. It is worth to mention that preferably, the selected audio file is an audio loop with constant tempo. Nevertheless, tempo and swing transformations can be applied to any sound material such as entire songs with unequal results.

2.4.5 Implementation

The implemented system in this sub-workpackage is a real-time system for rhythm manipulation of polyphonic audio signals. A rhythm analysis module extracts information of tempo and beat location. Based on this rhythm information, we apply different transformations: *Tempo*, *Swing*, *Meter* and *Accent*. This type of manipulation is generally referred as Content-based transformations. We address characteristics of the analysis and transformation algorithms. In addition, user interaction plays also an important role in this system. Tempo variations can be controlled either by tapping the rhythm with a MIDI interface or by using an external audio signal such as percussion or the voice as tempo control.



Figure 13. Graphical User Interface for the Groovator plugin

The system runs in real-time as a VST plug-in. There are two different implementations for the two operation modes: playback and streaming. The former is a VST-Instrument, while the latter is a VST-Effect. Next section describes the main differences of both implementations.

Although, the system was initially conceived only for playback mode, we implemented two versions of the Groovator: as VST-Instrument (playback) and as VST-Effect (streaming) for an easier integration in diverse audio applications. The former processes an audio file, while the latter processes an incoming audio stream. Although the transformations are the same in both implementations, the VST-Effect version presents some limitations as described below.

In streaming mode, tempo variations are limited to the length of an internal look-ahead buffer. In order to make the system robust, we implemented a self-recovering method that automatically initializes the buffer. It is achieved by changing the time scale factor smoothly until the buffer gets to its initial position (half the look-ahead buffer size). At that moment the user can change the tempo again. The recovering process is done in two-steps as shown in the Figure 13. Also, in streaming mode, the Groovator introduces an initial latency of 284ms due

to the time-scaling algorithm. We modified the time-scale algorithm, achieving a lower latency of 100ms. However, this modification implies a lower quality in the time-scaling algorithm, mostly perceived in transients.

For tempo conducting, where MIDI input is required, we implemented a function for adjusting the MIDI interface latency. After activating the calibration button, the user taps to a sequence of synthesized impulses. The system computes automatically the delay, which is used later for synchronization issues. Besides to the MIDI input, the system offers the possibility of synchronizing with a beat sequence using the Open Sound Control (OSC) protocol. On the GUI, user selects the listening port.

The controls of the Groovator GUI are separated in three groups Analysis, Control and Groove:

User Controls:

<i>Analysis:</i>	
BPM:	A text field indicates the current BPM value.
CALIBR:	This mode is used to estimate the latency of the MIDI interface, and is taken into account when the user control the playback speed through MIDI.
<i>Analysis:</i>	
AUTO/MANUAL:	In AUTO mode, the BPM value is taken from the automatic rhythm analysis. In MANUAL, the BPM value is specified through incoming MIDI messages. This mode is useful to apply <i>Groove</i> to an audio with unstable rhythm such as classical music
DOUBLE	This doubles the BPM value extracted by the automatic rhythm analysis in order to apply <i>Groove</i> with another resolution.
<i>Control:</i>	
MIDI:	We can control the playback speed through MIDI. All MIDI note messages sent to the the channel specified in the MIDI field are considered as <i>beats</i> . The BPM value and the playback speed is then modified according the the incoming MIDI messages.
OSC:	Similar to the MIDI, but instead of receivng MID Note messages, the <i>beats</i> are received through OSC in the specified port.
RESET:	It resets both the <i>Speed</i> and <i>Groove</i> sliders.
SPEED:	It modifies the playback speed with a range between 0.5 (faster) and 2.0 (slower). The <i>Speed</i> can be controlled either by changing the slider value or through incoming MIDI messages.
INERTIA:	When the playback speed is controlled through MIDI messages, we can avoid fast speed variations by increasing the <i>Inertia</i> value.
<i>Groove:</i>	
FACTOR:	It indicates the amount of time expansion/compression within a beat. In the default position (0.5) the timing is not altered.
OFFBEAT:	Currently the automatic rhythm analysis works in a eight-note resolution, meaning that we cannot predict the position of the <i>downbeat</i> . With the <i>Offbeat</i> button we can easilty switch between two type of groove modification.

2.4.6 Dissemination materials

2.4.6.1 *Scientific publications*

- [1]. Hazan, A. Performing expressive rhythms with BillaBoop voice-driven drum generator. Proceedings of 8th International Conference on Digital Audio Effects (DAFX 05), Madrid, Spain, 2005.
- [2]. Hazan, A. BillaBoop: real-time voice-driven drum generator. Proceedings of 118th Audio Engineering Society Convention (AES 118), Barcelona, Spain, 2005.
- [3]. Hazan, A. Towards automatic transcription of expressive oral percussive performances. *Proceedings of International Conference on Intelligent User Interfaces (IUI 2005)*, San Diego, CA, USA.
- [4]. Janer, J. Bonada, J. Jordà, S. "Groovator - an implementation of real-time rhythm transformations"; Proceedings of 121th Audio Engineering Society Convention; San Francisco, 2006

2.4.6.2 *Public presentations*

Janer, J; "Leisure and Voice Control"; Presentation at the 2nd Summer School in Sound and Music Computing, Barcelona; 23/07/2006

2.4.6.3 *Related PhDs*

Jordà, S. *Digital Lutherie: Crafting musical computers for new musics' performance and improvisation*, UPF Ph.D. Dissertation, 2005.

2.5 Player for the HiFi System

Responsible partner : UPF

2.5.1 Functional description

The SHF (Semantic HIFI) player component is responsible for reading soundfiles, decoding them and playing them back. Its functionality can be controlled over a UDP connection, making it possible to control it through custom controller components. The player's functionality also includes sound processing tasks, mainly via plugins, but also internally. Functionality like shuffle, repeat and playlist handling are built into the player.

2.5.2 Method description

The time-scaling algorithms used in the Player for the HiFi system have been already described in the section 2.4.2 of the Rhythm Transformation sub-workpackage.

2.5.3 Position over state-of-the-art

This sub-workpackage shares the position over the state-of the art with sub-workpackage 5.4 *Rhythm transformation*.

2.5.4 Benchmarks

For this sub-workpackage, user tests were carried out integrated in the final prototype of the HiFi system. No specific user evaluation for the module was required.

2.5.5 Implementation

2.5.5.1 *Input:*

The input component of the file-player is based on libsndfile (see <http://www.mega-nerd.com/libsndfile/> for more information) for accessing to most commonly used sound-file formats. This includes uncompressed formats such as .wav and .aiff as well as compressed formats (.ogg and .flac).

For decompression of MPEG2/Layer3 audio data, the MAD MPEG audio decoder is used (<http://www.underbit.com/products/mad/>). The player also supports digital reading from audio

CD's. AAC support could be included using the FAAD2 decoder, provided that a patent license is acquired.

(more on <http://www.vialicensing.com/products/mpeg4aac/licenseFAQ.html>).

Supported File-formats include:

through libsndfile:

WAV
AIFF/C
AU/SND
PAF Paris Audio File
IFF/SVX Amiga
Sphere Nist WAV
IRCAM SF
Creative VOC
Soundforge W64
MAT4 Octave
MAT5 Octave
PVF Portable Voice Format
Fasttracker 2 XI
HMM Toll Kit HTK
Apple CAF
FLAC
OGG

through additional libraries:

MPEG2/layer3
CDROM

not supported yet:

WMA
AAC
MPC
MusePack

2.5.5.2 Plugins:

The player also handles LADSPA plugins as well as plugins with asynchronous input and output buffers (used for time-stretch and re-sampling). The interface for LADSPA support is not yet included in this document, and is subject to be defined.

The API of the asynchronous plug-in system is not defined yet, we are still looking for a standard that can be reused (most plug-in systems require the input and output buffers to be of the same size).

2.5.5.3 Player Control

The Player component is controllable by Open Sound Control messages (OSC).

OSC is a protocol specifically designed for the communication between synthesizers and computers and due to its flexible naming scheme and parameter types ideal for the envisioned task.

OSC communicates via messages, which consist of an address and arguments. The address field can be structured hierarchically in order to access several components with the same features. In the following specification the first word in italic and starting with a “/” will be the address of the command. The arguments will be either of type “float”, “boolean” or “string”, which will be indicated by the first letter of the argument “b” for boolean and “f” for float, strings will be encapsulated in angle

brackets.

2.5.5.4 Filenames

In order to make access to different data containers possible, the filenames in the system are understood in an URL like syntax, where the player will be able to understand at least the following type of URL.

cdrom:number_of_track	The song is on the cdrom, track number number_of_track
file:<path_to_file>	path_to_file is a full path to a soundfile on the system (e.g. "/home/semantic/yesterday.mp3") upon missin ":" qualifier, file: is assumed

2.5.5.5 Command Specification

Note: Some of the “get...” commands for volume settings may not be needed, as they can be stored in the JAVA component. In this case the callback interface would only report the players position.

<i>Command</i>	<i>Arguments</i>	<i>Description</i>
<i>/play</i>		Play the first song of the current playlist
<i>/load</i>	<filename>	Load a song and play it
<i>/loadlist</i>	<filename>	Load a playlist
<i>/appendsong</i>	<filename>	Append the song indicated by <filename>
<i>/clearlist</i>		Clear the current playlist
<i>/stop</i>		Stop playback
<i>/pause</i>		Pause playback
<i>/next</i>		Go to next song
<i>/prev</i>		Go to previous song
<i>/skipto</i>	fTime	Skip to position fTime*10 milliseconds
<i>/ffwd</i>	fFrames	fast forward fFrames frames
<i>/frew</i>	fFrames	fast rewind fFrames frames
<i>/setvol</i>	fVol	set volume (0-100)
<i>/getvol</i>		send volume information through return connection.
<i>/setbass</i>	fBass	set bass volume (0-100)
<i>/getbass</i>		send bass volume value through return connection.
<i>/setmiddle</i>	fMiddle	set middle volume (0-100)
<i>/getmiddle</i>		send middle volume value through return connection.
<i>/settreble</i>	fTreble	set treble volume
<i>/gettreble</i>		send treble volume value through return connection.
<i>/setrepeat</i>	bRep	set repeat state
<i>/getrepeat</i>	bRep	send value of repeat flag through return connection.

<i>Command</i>	<i>Arguments</i>	Description
<i>/setshuffle</i>	bShuffle	set shuffle
<i>/getshuffle</i>		send value of shuffle flag through return connection.
<i>/setPosition</i>	fInterval	send position information through return connection at a regular interval of fInterval milliseconds.

2.5.5.6 *Callback Messages from the Player*

Command	Arguments	Description
/volume	fVolume	the current volume is fVolume
/bass	fBass	the current bass volume is fBass
/middle	fMid	the current middle volume is fMiddle
/treble	fTreble	the current treble volume is fTreble
/repeat	fRepeat	the current value of the repeat flag
/shuffle	fShuffle	the value of the Shuffle flag
/position	fPosition	the current position in 10 ms entities

2.5.5.7 *Future Features of the Player*

Future features of the player might include the following:

- Information about the content of Audio CD-s, such as number of tracks and track lengths. This information might be used for a CD database lookup.
- Additional flags for contents sensitive playback such as automatic cross-fade using the BPM information.
- API for handling plugins and plug-in control

2.5.6 Dissemination materials

All dissemination within this sub-workpackage has already been described in section 2.4.6 of sub-workpackage 5.4 *Rhythm transformation*.

2.6 Score Follower

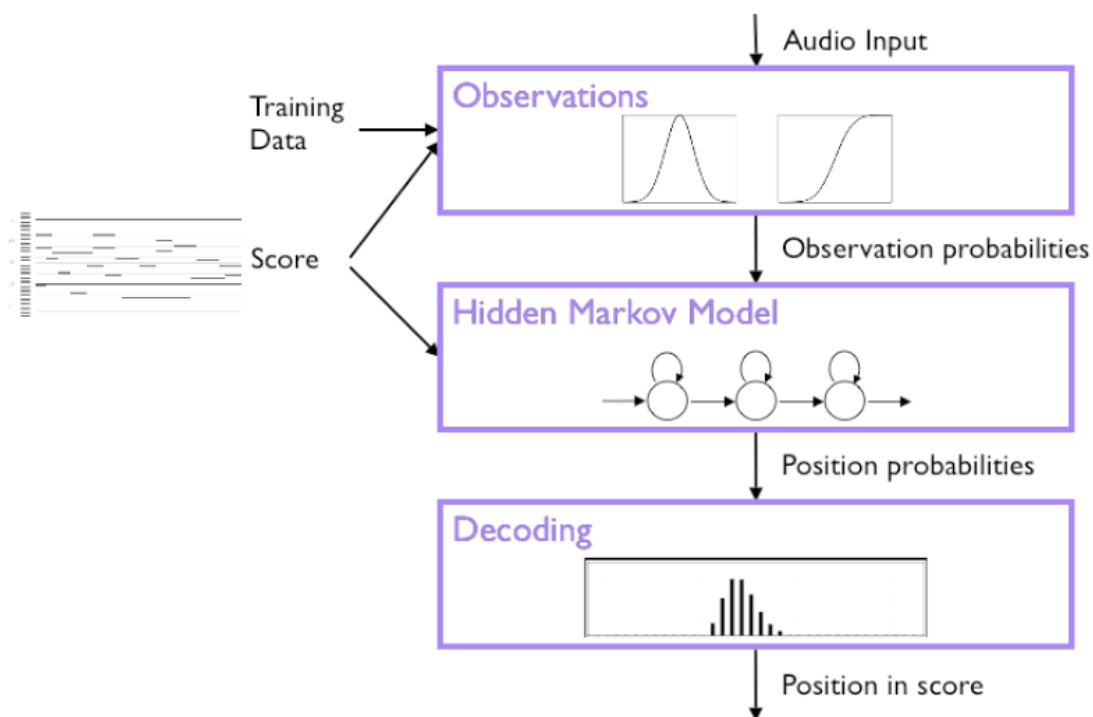
Responsible partner : IRCAM-ATR

2.6.1 Functional description

Score following is the key to an interaction with a written score/song based on the metaphor of a performer with an accompanist or band. While for many Pop and Rock music songs the usual karaoke setup of a performer singing along with a simple recording of the accompaniment is sufficient, especially for a repertoire of classical and jazz songs, ballads and traditionals, the synchronization of the accompaniment to the performer is indispensable.

The Score Following Player designed in the framework of this project adapts and improves a score following algorithm originally developed for contemporary music performance in order to create an easy to use and robust automatic accompaniment application accepting monophonic audio and MIDI input from the performer. The included audio and MIDI following modules use the same core algorithm based on Hidden Markov Models (HMM).

The following figure shows an overview of the follower algorithm:



2.6.2 Position over state-of-the-art

Score following is the real-time alignment of a known musical score to the audio signal produced by a musician playing this score in order to synchronise the electronic part of the music to the performer, leaving him with all possibilities of expressive performance. It now has a history of about twenty years as a research and musical topic.

The research on score following was initiated by Roger Dannenberg ⁰, Barry Vercoe and Lawrence Beauregard as soon as 1983 ⁰. It was continued by Miller Puckette and Philippe

Manoury 00, and other research groups 0. Since 1999, the Real Time Applications team (ATR) continues work on score following as their priority project 0000.

For an introduction and state of the art on score following and details of the system developed by IRCAM's Real-Time Applications team, see 00. A review of past attempts in score following literature, focusing on the adaptability and learning aspects of the algorithms, specially of importance for our work, is given in 0 and 0. Robust score following is still an open problem in the computer music field.

During the SemanticHifi project, the training algorithm for the HMM core algorithm of the score follower was improved to make the system usable by untrained singers. Training in this context is to adapt the parameters of the following algorithm to a certain instrument and to a certain style of performance. A novel automatic discriminative training algorithm was introduced, which, unlike classical methods for HMM-training, is not concerned with modelling the music signal but with correctly choosing the sequence of music events that, was performed.

The learning for the Score following Player prototype application was performed on a database of recordings of various male and female inexperienced singers. After the automatic offline learning, the system provides significantly improved alignment off the score following.

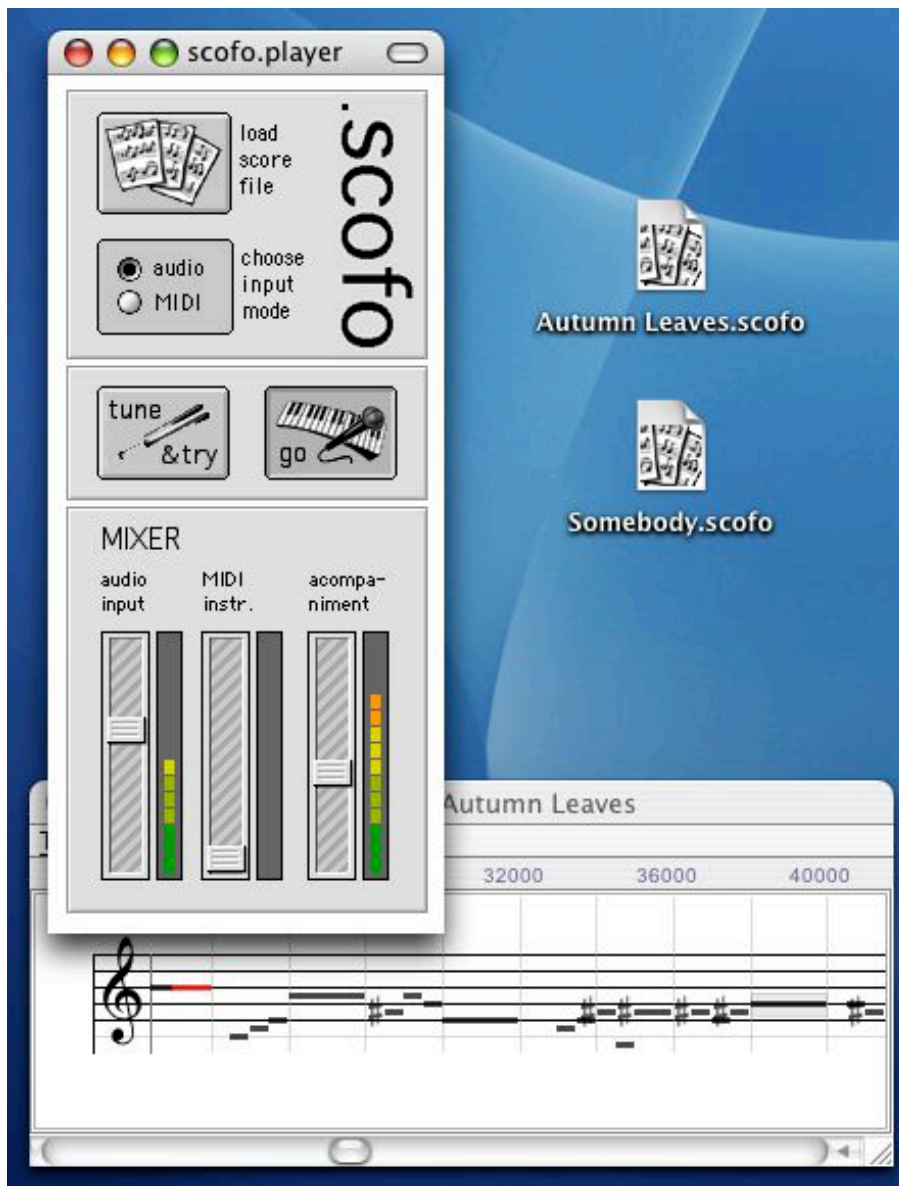
- [1] Roger B. Dannenberg. An On-Line Algorithm for Real-Time Accompaniment. In *Proceedings of the ICMC*, pages 193--198, 1984.
- [2] Barry Vercoe. The Synthetic Performer in the Context of Live Performance. In *Proceedings of the ICMC*, pages 199--200, 1984.
- [3] Barry Vercoe and Miller Puckette. Synthetic Rehearsal: Training the Synthetic Performer. In *Proceedings of the ICMC*, pages 275--278, 1985.
- [4] Puckette, M.S. (1995). Score following using the sung voice. In *Proceedings, International Computer Music Conference*, 1995.
- [5] Cano, P., Loscos, A. and Bonada, J. (1999). Score-Performance Matching using HMMs. In *Proceedings of the International Computer Music Conference*, 1999.
- [6] Orio Nicola, Déchelle François, Score Following Using Spectral Analysis and Hidden Markov Models. *ICMC: International Computer Music Conference*. La Havane : 2001
- [7] Orio Nicola, Lemouton Serge, Schwarz Diemo, Schnell Norbert, Score Following: State of the Art and New Developments. *New Interfaces for Musical Expression (NIME)*. Montreal : Mai 2003
- [8] Kaprykowsky Hagen, Time modeling in Hidden Markov Models. (in the framework of an internship at IRCAM) Universität Karlsruhe, Allemagne, 2004.
- [9] Arshia Cont, Improvement of Observation Modeling for Score Following, Memoire de stage de DEA ATIAM annee 2003-2004 Universite Pierre et Marie Curie, PARIS VI, Paris 2004
- [10] Schwarz Diemo, Orio Nicola, Schnell Norbert, Robust Polyphonic Midi Score Following with Hidden Markov Models. *International Computer Music Conference (ICMC)*. Miami : Novembre 2004
- [11] Cont Arshia, Schwarz Diemo, Schnell Norbert, Training IRCAM's Score Follower. *AAAI Symposium 2004 Style and Meaning in Language, Art, Music, and Design*. Washington : Octobre 2004
- [12] Cont Arshia, Diemo Schwarz, Schnell Norbert, Training Ircam's Score Follower. *IEEE International Conference on Acoustics, Speech, and Signal Processing*. Philadelphia : Mars 2005
- [13] Schwarz Diemo, Cont Arshia, Schnell Norbert, From Boulez to Ballads: Training Ircam's Score Follower. *International Computer Music Conference (ICMC)*. Barcelona : Septembre 2005

2.6.3 Implementation

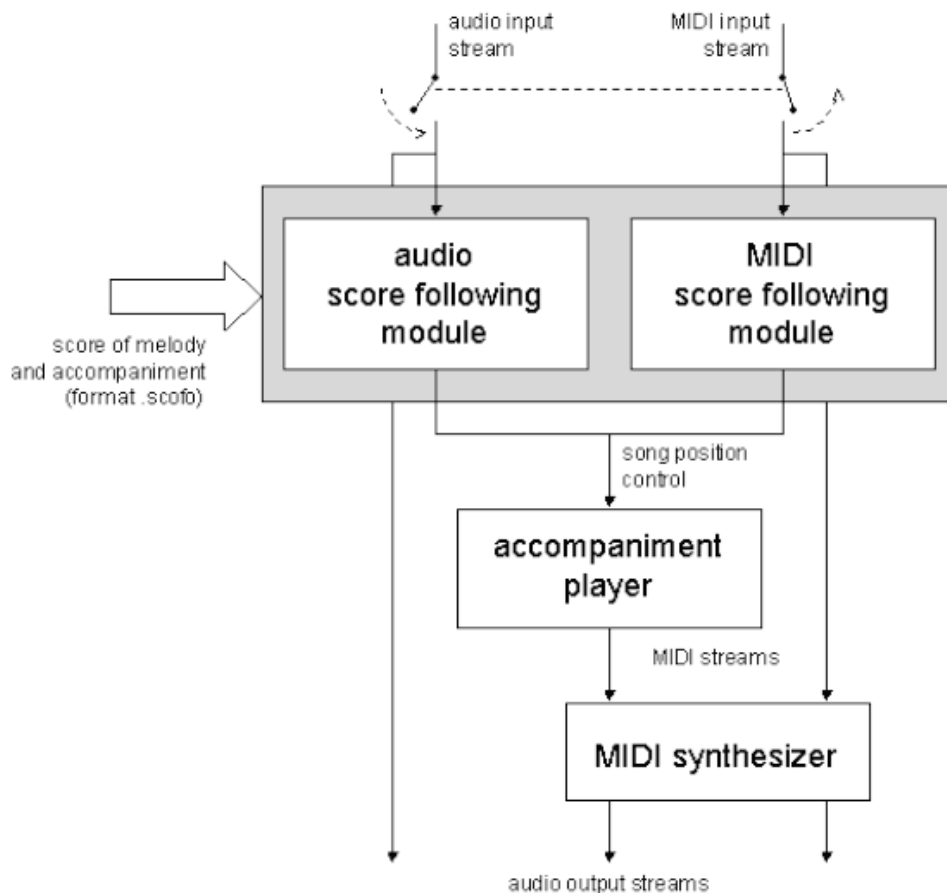
In the framework of the SemanticHifi project we developed a demo application that allows users of a Hifi system of the future to interact with the system and its music collection.

The score following application developed for this live demonstration (see the screen shot below) allows for automatic synchronization of a pre-composed accompaniment to a hobby

singer. The melody and the accompanying chords of a ballad such as *Autumn Leaves* are chosen and loaded into the application. During the performance, the solo melody is sung into a microphone connected to the system, which plays the accompaniment precisely synchronized to the singer's performance.



The application mainly consists of the score following module and an accompaniment module connected to a General MIDI synthesizer, as shown in the software architecture diagram below. The score following module, receiving the audio input from the singer or MIDI input played from an external keyboard, continuously estimates the current position in the performed song, output as a cue number. This output is used by the accompaniment module to look up the chord sequence associated to the cue, which is sent to the MIDI synthesizer. The accompaniment module can be schematized as a finite state machine advancing in the pre-composed sequence of chords driven by the output of the score following module and an internal timer. The internal timer is adjusted to the singer's rhythm and allows advancing in the chord sequence in the case that the singer pauses (according to the score or by error).



In the case that the singer sings out of tune or an unexpected melody, the score following module adjusts as well as possible the output position in the song by waiting and advancing. The module turns out to perform robust musical accompaniment in a number of situations usually judged as difficult to handle for automatic accompaniment systems, such as the singer deviating from the score and singing out-of-tune.

Musical material for the score following player, including the solo voice and the accompaniment, can be easily created. A song file format was defined based on the MIDI Standard file format. The advantage of this approach is the possibility to easily create and edit songs for the Score Following Player in any sequencer software. A song file defines the solo voice to follow and the accompaniment in two different MIDI tracks.

2.6.4 Dissemination materials

2.6.4.1 *Scientific publications*

- [1] Schwarz Diemo, Orio Nicola, Schnell Norbert, Robust Polyphonic Midi Score Following with Hidden Markov Models. *International Computer Music Conference (ICMC)*. Miami : Novembre 2004
- [2] Cont Arshia, Schwarz Diemo, Schnell Norbert, Training IRCAM's Score Follower. *AAAI Symposium 2004 Style and Meaning in Language, Art, Music, and Design*. Washington : Octobre 2004
- [3] Cont Arshia, Diemo Schwarz, Schnell Norbert, Training Ircam's Score Follower. *IEEE International Conference on Acoustics, Speech, and Signal Processing*. Philadelphia : Mars 2005

[4] Schwarz Diemo, Cont Arshia, Schnell Norbert, From Boulez to Ballads: Training Ircam's Score Follower. *International Computer Music Conference (ICMC)*. Barcelona : Septembre 2005

2.6.4.2 Related PhDs

Cont Arshia, Musical Anticipation, Ircam-STMS and University of California at San Diego (UCSD), ongoing thesis work.