UNIVERSITY OF CALIFORNIA

Santa Barbara

Adapt: A Networkable Plug-in Host for Dynamic Creation of Real-Time Adaptive Digital Audio Effects

A Project submitted in partial satisfaction of the

requirements for the degree of Master of Science

in Media Arts and Technologies

by

Matthew E. Stabile

Committee in charge:

Curtis Roads, Chair

Stephen Travis Pope

Matthew Turk

JoAnn Kuchera-Morin

March 2010

Adapt: A Networkable Plug-in Host for Dynamic Creation of Real-Time

Adaptive Digital Audio Effects

by

Matthew E. Stabile

ABSTRACT

Adapt: A Networkable Plug-in Host for Dynamic Creation of Real-Time Adaptive Digital Audio Effects

by

Matthew E. Stabile

Adapt is an audio effects plug-in host that allows for dynamic creation and exploration of Adaptive Digital Audio Effects (A-DAFx). An adaptive effect is created when extracted features from an audio signal are used to modulate any parameter of an audio effect algorithm. Well-known examples include a compressor or the pitch-controlled "auto-tune" effect. These are classified as auto-adaptive effects because the adaptive control signal is generated based on analysis of the same audio signal that feeds the effects algorithm; if the analysis is instead performed on a secondary audio signal, the effect is defined as external-adaptive [1].

Adapt allows for dynamic creation of auto-adaptive and external-adaptive effects by providing a performance interface which includes two separate sound sources, a bank of real-time spectral and time-domain audio analysis modules, a bank of user-defined effects plug-ins and a bank of mapping modules for creating continuous mappings between analysis control signals and effects parameters.

Beyond the many configurations possible within a single instance of the program, Adapt is specifically designed to allow creation of A-DAFx between two live performers processing their instruments on separate computers. When two instances of Adapt are

connected over a network, Open Sound Control is used to share all analysis module control signals in real-time. Each performer is then able to create mappings using both their local analysis control signals and the remote performer's control signals, allowing dynamic creation of complex and novel A-DAFx.

TABLE OF CONTENTS

I. Introduction

Digital audio effects are widely used tools in live musical performance and music production, both in the studio and at home. In recent years, a new classification of effects known as Adaptive Digital Audio Effects (A-DAFx) has emerged, which involves using extracted features from an audio signal to modulate any parameter of an audio effect algorithm. Specific examples of A-DAFx include dynamics effect processors (compressor, expander, limiter, etc.) and the increasingly popular pitch-controlled effects such as "auto-tune" or the intelligent harmonizer [1]. The three key components of any adaptive effect include the audio analysis algorithm, the effect algorithm and the mapping functions used for modification of the adaptive control signal.

When generalized into these separate components, it is apparent that an incredible amount of possibilities exist for creation of new A-DAFx since the output of *any* existing audio analysis algorithm can be linked to *any* existing audio effect parameter. In addition, for every link that is defined, unique mapping functions may be applied which will yield widely varying results for any given input signal. Of course, for the vast amount of theoretically possible combinations, only a small subset is likely to yield musically pleasing results. With this in mind, it is clear that the process of exploring new A-DAFx algorithms could greatly benefit from a modular environment that contains a large selection of real-time analysis and effects modules and allows for dynamic mapping between the two.

A number of modular plug-in hosts exist which partially fulfill this need by exposing the parameters of effects plug-ins for modulation; however, they are for the most part focused on music synthesis and parameter automation, and each lack a

significant selection of real-time audio analysis modules and dynamic mapping utilities. Adapt provides this functionality by combining together the relevant features of these modular plug-in hosts, a collection of state-of-the-art real-time audio analysis algorithms and a selection of mapping utilities drawing from research involving gestural control of sound synthesis.

A significant contribution to highlight is the inclusion of a library of spectral-domain audio analysis algorithms. Following the release of the MPEG-7 standard, researchers in the field of Music Information Retrieval (MIR) have made significant advancements in developing real-time spectral-domain audio analysis algorithms for the extraction of semantic content descriptors [2]. These semantic descriptors, such as brightness, harmonicity, and noisiness, are in widespread use for music classification and recommendation systems; however, they have just begun to be used for music composition and synthesis and their direct use for A-DAFx is rare [3].

Section II begins by defining common A-DAFx terminology, providing examples of existing A-DAFx and describing other techniques relevant to A-DAFx. Next related research specifically involving A-DAFx is presented, as well as relevant research in the fields of MIR and gestural control of sound synthesis. Related effects processing software applications are also discussed. Section III provides insight into the motivation and tool selection process for the project. Section IV provides an overview of the design and implementation of Adapt. Sections V thru VII discuss the results of this implementation, future work for the project and conclusions.

II. Related Work

A. A-DAFx Terminology

The term “Adaptive Digital Audio Effects” was formally introduced by Verfaille et al. in [1]. Although the general classification was new, various forms of A-DAFx have been around for quite some time and are also commonly referred to as Intelligent Digital Audio Effects [4], content-aware effects, or content-based transformations [5]. What they all have in common is the use of extracted sound features as an adaptive control for parameters of audio effects. Traditionally, the controls of audio effects stay constant with time or are manually modified by the user via a knob, slider or foot pedal. If the parameters are automatically modulated by a signal, it is typically by a low-frequency oscillator (LFO) or a pre-programmed automation curve. The motivation for using an adaptive control comes from the fact that extracted sound features are inherently related to a musical gesture, which provides a meaningful and intelligible control [1].

Different forms of A-DAFx are defined depending on what signal is used for the feature extraction. The effect is defined as auto-adaptive if the adaptive control signal is generated based on analysis of the same audio signal that feeds the effects algorithm. If the audio signal being analyzed is different than the audio signal feeding the effects algorithm, the effect is defined as external-adaptive. Each of those are feed-forward A-DAFx, and a feedback adaptive effect extracts features from the output signal [1]. If two or more external-adaptive effects are combined, the effect is defined as cross-adaptive since each effect algorithm is being modulated by a control signal derived from the other signal.

It follows that all existing adaptive effects algorithms may be classified and

grouped according to these forms. The following are examples of well-known effects that are in fact adaptive effects. The “auto-wah” or “envelope filter” is an auto-adaptive effect that uses an envelope follower for analysis, with the input audio envelope directly controlling the cutoff frequency or resonance of a filter. Dynamics effects processors are also auto-adaptive effects utilizing an envelope follower, with the control signal adjusting the volume of the input signal in various ways. A compressor can also be an external-adaptive effect through the use of a “side-chain” input, which will be explained in detail in the next section. The “auto-tune” and intelligent harmonizer effects are auto-adaptive with a fundamental frequency estimation algorithm performing the analysis.

The extremely popular “Talk Box” effect, which has been in use since the 1970’s, can be classified as an external-adaptive effect where the frequency content of the instrument’s sound is filtered and modulated by the shape of the performer’s mouth before amplification. The traditional “Talk Box” is an interesting example since it does not use analog or digital circuitry for the analysis and modulation, but rather the physical characteristics of how the mouth processes the human voice and audio input signal. The “Talk Box” effect is often confused with the extremely common vocoder technique, which is also an example of an external-adaptive effect where the spectral envelope of one audio signal modifies the spectral envelope of another audio signal.

B. Existing Techniques

In addition to these existing examples of adaptive effects, a number of related modulation techniques exist. These include matrix modulation from the world of analog synthesizers, as well as two techniques often used in music production known as side-

chaining and parameter automation.

Matrix Modulation

Modular analog synthesizer designers pioneered the use of continuous control signals for modulation of effects and synthesis parameters. The majority of analog synthesizers include several modulation sources, usually including one or more LFOs, envelope generators, and analog step-sequencers. These control signals are used to modulate voltage-controlled oscillators (VCOs), voltage-controlled filters (VCFs) and voltage-controlled amplifiers (VCAs). The modulation sources are typically routed to these destinations via a modulation matrix or by manually connecting them through the use of patch cables; additional controls are often provided for adjusting the amount or depth of modulation. Figure 1 shows a vintage analog synthesizer featuring a modulation matrix. The oscillators and modulation sources are listed on the left-hand side of the matrix, and are connected to the destinations listed along the top by inserting colored pins into the holes of the matrix. A typical modulation effect routing would connect an envelope-generator to the cutoff frequency of a low-pass filter, adding texture to the audio signal produced by the synthesizers' oscillators.

Figure 1a: EMS VCS-3 Analog Synth

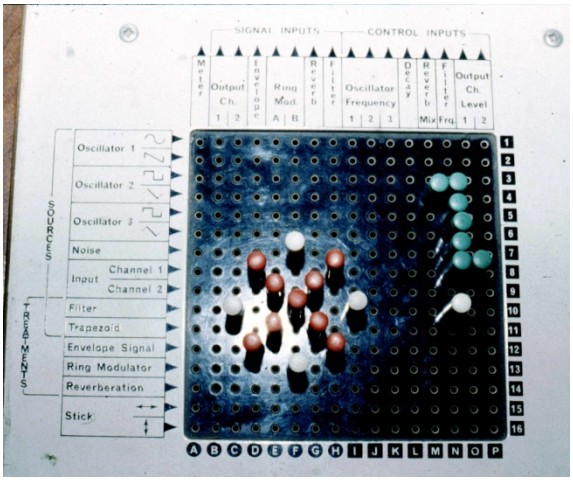

Figure 1b: VCS-3 Modulation Matrix

Side-Chaining

There is a technique known as side-chaining that is used extensively in modern music production and is essentially an external-adaptive effect that exclusively uses envelope following as the analysis algorithm. Side-chaining is most often associated with compressors, as many hardware and software compressors have a "side-chain" input, which is intended for an auxiliary audio input to be used as a control signal for the amount of compression to apply. The action a side-chain compressor performs is commonly referred to as "ducking" because whenever the secondary audio input reaches a certain volume level, the audio source being processed through the compressor "ducks" out of the way by reducing in volume. The technique can clearly be heard on the radio whenever a DJ begins to speak and the background music automatically reduces in volume.

Side-chain inputs are common in many software plug-ins and side-chain routing is available in most popular Digital Audio Workstations (DAWs). Side-chaining is an adaptive technique that has gained widespread use and is a good indicator as to the potential for external-adaptive effects that use more sophisticated analysis algorithms.

Parameter Automation

Parameter automation is another widely used production technique and is a feature that can be found in many modern DAWs. Parameter automation is the recording of manual adjustments that are made to effects parameters throughout the duration of a track. DAWs achieve this by recording an automation curve that is then displayed next to or on top of the track that the audio effect is applied to. Once recorded, the automation curve may be visually adjusted for synchronization to musical events. The end result of

parameter automation is thus very similar to an adaptive effect, with the difference being that a human, rather than an algorithm, is doing the musical analysis. This technique could indeed be extended by providing the option of using an analysis algorithm for initial recording of the automation curve.

C. A-DAFx Research

Substantial research projects specifically focused on A-DAFx have only become prevalent in the past decade, following the initial classification of A-DAFx by Verfaille et al. in [16] and the general classification and framework for creation of A-DAFx provided in [1]. This general classification systematically introduced the various A-DAFx forms, generalized mapping strategies, and the classification of adaptive effects and analysis algorithms using perceptual attributes. A selection of novel adaptive effects and their implementations were also proposed in [1], including adaptive equalization, adaptive tremolo, adaptive time warping, adaptive pitch-shifting, automatic vibrato, spectral warping and an adaptive granular delay.

In addition to the initial general classification of A-DAFx, there exist several other research efforts in the field, which are focused on specific kinds of adaptive effects (these groups tend to prefer the term Intelligent Digital Audio Effects). One of the larger efforts is on the topic of automatic mixing, which is a collection of tools for automation of a variety of complex mixing tasks. Automatic mixing requires analyzing the content of multiple channels of audio at a time, each with respect to each other. Examples include automatic panning, automatic gain maximization and feedback prevention and automatic faders; these are all examples of cross-adaptive effects, also known as inter-channel

dependent effects or multi-input/multi-output effects [6].

Other research involves the use of a beat-tracking system to create tempo-synchronous and beat-synchronous audio effects. In these the extracted tempo of the input signal is used to automatically adjust the delay time as the performance progresses. Beat-synchronous low-frequency oscillator (LFO) effects are created by automatically synchronizing the rate of the LFO to the detected beats of the input signal. This allows for creation of effects such as a beat-synchronous tremolo, vibrato or auto-wah [7].

D. Real-Time Audio Analysis

The audio analysis algorithm is the essence of an adaptive effect, since it is in fact how the adaptive control is derived. In particular, it is real-time audio analysis algorithms that are of special interest to this project because those are what enable the use of A-DAFx in live musical performance.[1]

In order to extract the characteristics of higher-level musical gestures from an audio source, an understanding of the relationship between known signal processing techniques and the gestures is needed. The high-level features of any musical audio signal are most often classified using a perceptual categorization; namely loudness, pitch, time, space and timbre. Each perceptual attribute has a defined relationship with one or more low-level sound features, which are what we can extract using real-time signal processing techniques. A selection of these relationships, as proposed by Verfaille et al., are shown in Figure 2 with the extraction techniques on the left side and the higher-level perceptual

[1] It should be noted here that the term "real-time" does not denote that the calculations are instantaneous; they are in fact delayed by a fixed number of samples since a block-by-block computational approach is used. The control rate is a fixed amount lower than the audio sampling rate, but by an amount that is still within acceptable latency for human perception.

attributes on the right side. This non-exhaustive set of features is drawn from those used for timbre space description as defined in the MPEG-7 multimedia content description standard [1]. The algorithms and attributes presented in Figure 2 are the analysis building blocks to be used for creation of A-DAFx in Adapt.

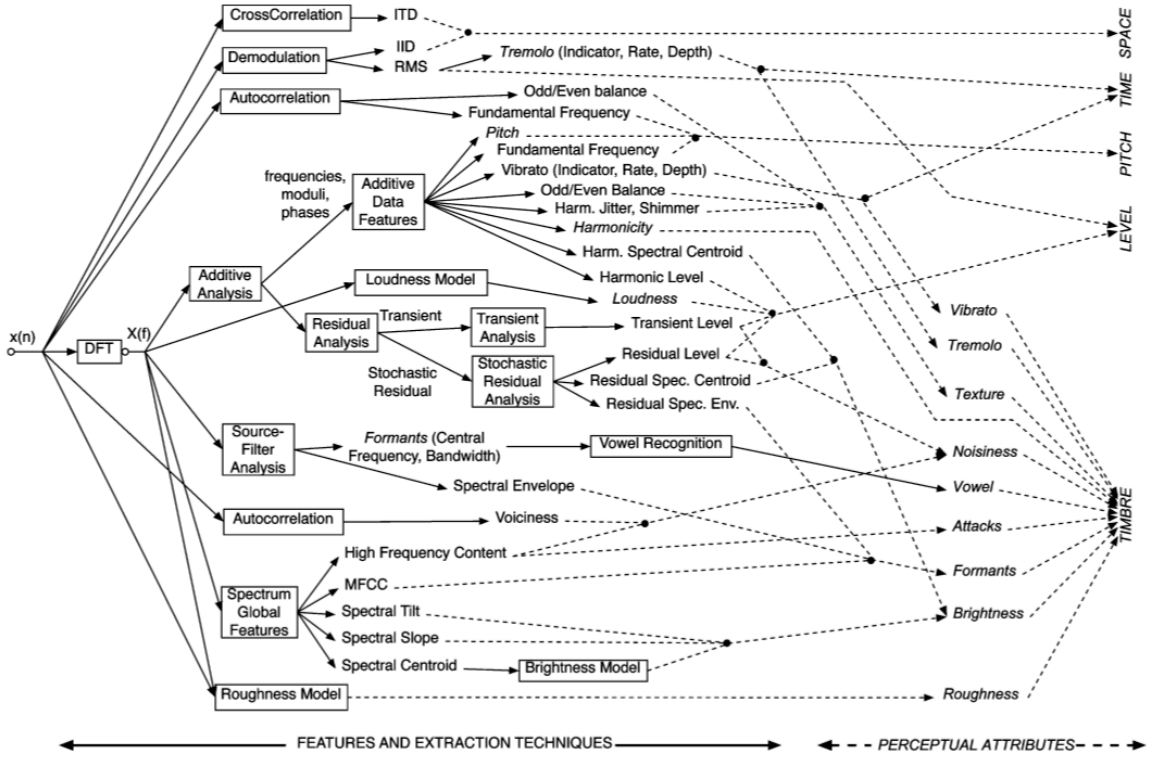

Figure 2: Feature extraction techniques and perceptual attributes.

The passing of the MPEG-7 multimedia content description standard has greatly increased the amount of MIR research going into algorithms for extracting semantic content descriptors from audio data. The main motivation is for gathering data to assist in indexing and browsing very large sound databases, but some musicians and researchers have begun to use these descriptors for music composition and synthesis [3].

The descriptors of particular interest are computed by spectral-domain algorithms and until recently the calculation of large amounts of spectral data has been too expensive to perform in real-time. Researchers at IRCAM have developed a library of Max/MSP

objects named Zsa.Descriptors, which uses a modular approach to allow for efficient computation of multiple descriptors in real-time [3]. The library provides many of the extraction techniques that describe attributes related to timbre as illustrated in Figure 2; the use of these in Adapt is described in detail in Section IV.

E. Mapping Strategies

Before the output of an analysis algorithm can be directly mapped to a parameter of an audio effect, it must first undergo signal conditioning to ensure that the proper ranges are satisfied or to create a more dynamic response. Mapping strategies are of great interest to researchers concerned with gestural control of sound synthesis, which shares many of the same needs as the conditioning for an adaptive control. An open-source project of particular interest is the Digital Orchestra Toolbox (DOT), which is a collection of Max/MSP utilities needed for common mapping tasks [9]. The DOT mapping utilities employed for Adapt include range scaling using linear or non-linear transfer functions, automatic detection of minimum and maximum ranges limits, logarithmic smoothing filters, configurable range clipping and value ramping.

Relevant software that has been developed using the DOT utilities is the DOT Mapping Tools, which is a framework and GUI for mapping controller gestures to sound synthesis parameters [10]. The functionality provided by the mappings tools is very similar to that needed for creation of A-DAFx. The control signals in Mapping Tools come from gestures captured by physical input devices, whereas A-DAFx control signals come from analysis of audio signals. The control signals in mapping tools get mapped to sound synthesis parameters, whereas A-DAFx analysis signals get mapped to effects

parameters. While the two have much in common from a tools standpoint, the end musical result is much different.

F. Related Software

A number of commercial software programs exist that provide a portion of the functionality needed for dynamic real-time creation of A-DAFx. Most of them are modular patcher-based environments, which are focused on sound synthesis and composition, but also include VST hosting capabilities and live signal input processing. While certain A-DAFx can be created within them, they each lack a significant selection of real-time audio analysis modules and dynamic mapping utilities.

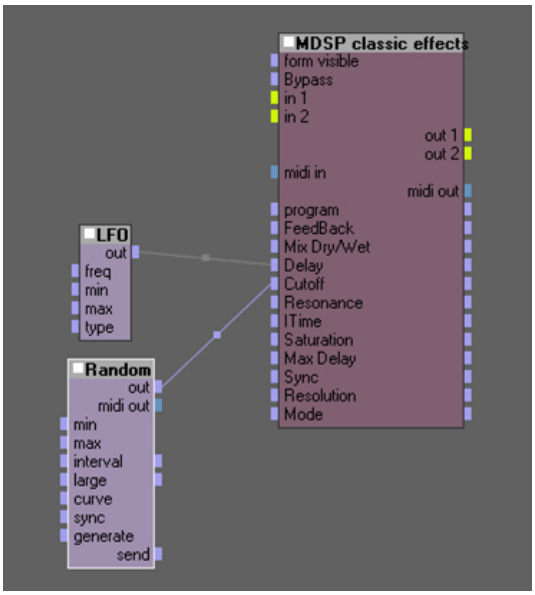

Figure 3: Usine Patch with VST Module

Sensomusic's Usine is a commercial application designed for live sampling, sound design, and effects processing.[2] It provides a patching mode where VST plug-ins may be loaded in, creating a module with all of the effects parameters exposed (see Figure 3).

[2] http://www.sensomusic.com/usine/

Any signal can then be connected with a patch cable and the parameters will be modulated in real-time by that signal. Figure 3 shows an example with an LFO and random signal generator connected to a VST module. Usine includes three analysis modules, which are an envelope follower, a peak detector (uses a threshold on the envelope follower) and a tempo detector. A handful of A-DAFx may be created using those, but since Usine has a fixed number of high-level modules, the mapping options it provides are very minimal and further exploration is prevented.

Ross Bencina's Audio Mulch is another commercial patcher-based environment, which is very similar to Usine and provides high-level modules designed for live electronic music performance, composition and sound design [11]. Audio Mulch is also a fully functional VST host and is capable of loading VSTs into modules with their parameters exposed for modulation. Audio Mulch has extensive parameter automation capabilities; however, it does not include any audio analysis modules beyond an envelope follower and thus creation of A-DAFx is very limited.

Another program that allows for modulation of effects parameters is Native Instruments' Guitar Rig.[3] Guitar Rig in fact has a "modifier" bank, but the only analysis module included is an envelope follower, and the rest are LFOs, envelope generators, and sequencers. Guitar Rig is not a plug-in host, so the user is limited to the effects that come with the program.

Also of note is a program called Circle, which is a modular software synthesizer by Future Audio Workshop. Although it is a modular software synthesizer, and thus does not include effects processing for live input signals or recorded sounds, it's extremely intuitive approach to modulation routing is of special interest. In essence, the modulation

[3] http://www.native-instruments.com/#/en/products/guitar/guitar-rig-4-pro/

routing is performed the same as in an analog modular synthesizer that uses patch cables (rather than a modulation matrix) for connecting modulation sources to VCOs and VCFs. The difference is that rather than using graphical representations of patch cables, each of Circle's modulators has a unique color assigned to it and connections are made by dragging a small circle of that color around the screen and dropping it onto any controllable parameter on the GUI (Figure 4). The modulation effect can also be previewed before assignment by simply hovering the glowing colored circle over the desired parameter. The color-coding allows all modulation sources and destinations to easily and instantly be identified and reconfigured in real-time. This type of modulation routing strategy would be highly useful for exploring new A-DAFx algorithms and is a possible future direction for Adapt.

Figure 4: Screen shot of Circle's GUI

Cycling 74's Max/MSP is a patcher-based programming environment, which provides all of the VST parameter modulation capabilities as USine and Audio Mulch, but also provides extensive low-level functions for creating custom mapping modules.[4] In addition, a large amount of audio analysis modules already exist and a C-based API allows for unlimited extensibility and addition of new analysis algorithms. Any program developed in the Max/MSP environment can also be compiled into a fully stand-alone program, so for these reasons Max/MSP was chosen as a development platform for the Adapt software. Further details regarding the design of Adapt and the functionality provided by Max/MSP follow in the next section.

III. Motivation and Tools

The initial motivation for Adapt came out of research involving the creation of single instances of adaptive effect plug-ins. An adaptive effect plug-in requires the three main components of any adaptive effect, which are an analysis algorithm, an effect algorithm and the mapping functions which link the two together. Once all of these are hard-coded into a plug-in architecture, it becomes apparent that all of the controls for the analysis algorithm and mapping functions are needed on the GUI for proper exploration of the capabilities of the effect. With these in place, the plug-in is now a completed auto-adaptive effect; if an external-adaptive effect is desired, then a side-chain input must be added and the plug-in must be used in a host which supports side-chain signal routing.

Next, the desire comes to find out how that particular analysis algorithm sounds with a different effect algorithm, or how a different analysis algorithm will sound with that same effects algorithm. At this point, it is clear that separating the analysis algorithm

[4] http://cycling74.com/products/maxmspjitter/

and mapping functions from the effect algorithm will save a great deal of programming time and allow for much faster proto-typing of new A-DAFx algorithms. Also, since such a large variety of free and commercial effects plug-in already exist, the development time can be isolated to creating new analysis algorithms and mapping schemes, rather than packaging together single effects and analysis algorithms for each sonic experiment.

With the decision to isolate the analysis algorithms, a number of different approaches may be taken. Since it is desirable to utilize the large amount of effects plug-ins readily available, it follows to focus on the creation of analysis plug-ins for use within an environment capable of hosting plug-ins. The environment where plug-ins are most commonly used is a Digital Audio Workstation, where they are inserted on individual tracks for adding effects to a pre-recorded track. Since the initial motivation for this project lies in the domain of exploring the use of A-DAFx in a live performance situation, the DAW environment was not chosen as a suitable host for A-DAFx research. However, a suite of analysis plug-ins and a general solution for internal plug-in communication inside of any given DAW would be a very interesting and useful research project in itself.

For A-DAFx research involving live performance, there exists another category of plug-in hosts that are concerned with live sampling, effects processing and music synthesis. These were discussed in section II, where it was elucidated that after an evaluation of all available hosts, Cycling 74's Max/MSP was chosen as a development platform due to a large number of relevant programming utilities and modules, coupled with sufficient extensibility.

Adapt was written using an open-source framework called Jamoma[5], which is a

[5] http://www.jamoma.org

new standard for developing high-level modules in the Max/MSP Environment [12]. Jamoma was chosen for this project because of its elegant use of Open Sound Control (OSC)[6] for internal and inter-module communication [13]. Every parameter in each analysis and effect module within Adapt has a unique OSC address, which is what makes the sophisticated mapping possibilities and networking options possible.

The Jamoma framework is completely modular and implements a Model-View-Controller (MVC) strategy [17]. Jamoma defines standards for each module's messages, parameters, GUI dimensions, state management, parametric control and automation capabilities [12]. Once a Jamoma module is created it can automatically communicate via OSC to all other modules, so this framework proved to be a perfect fit for the dynamic mapping and flexibility that Adapt requires.

Figure 5: Example of a Jamoma module in Adapt

[6] http://opensoundcontrol.org/

IV. Adapt Program Implementation

This section describes all of the modules and features of Adapt. It begins with the audio routing options, and then describes the analysis bank, the mapping bank and the effects bank in detail.

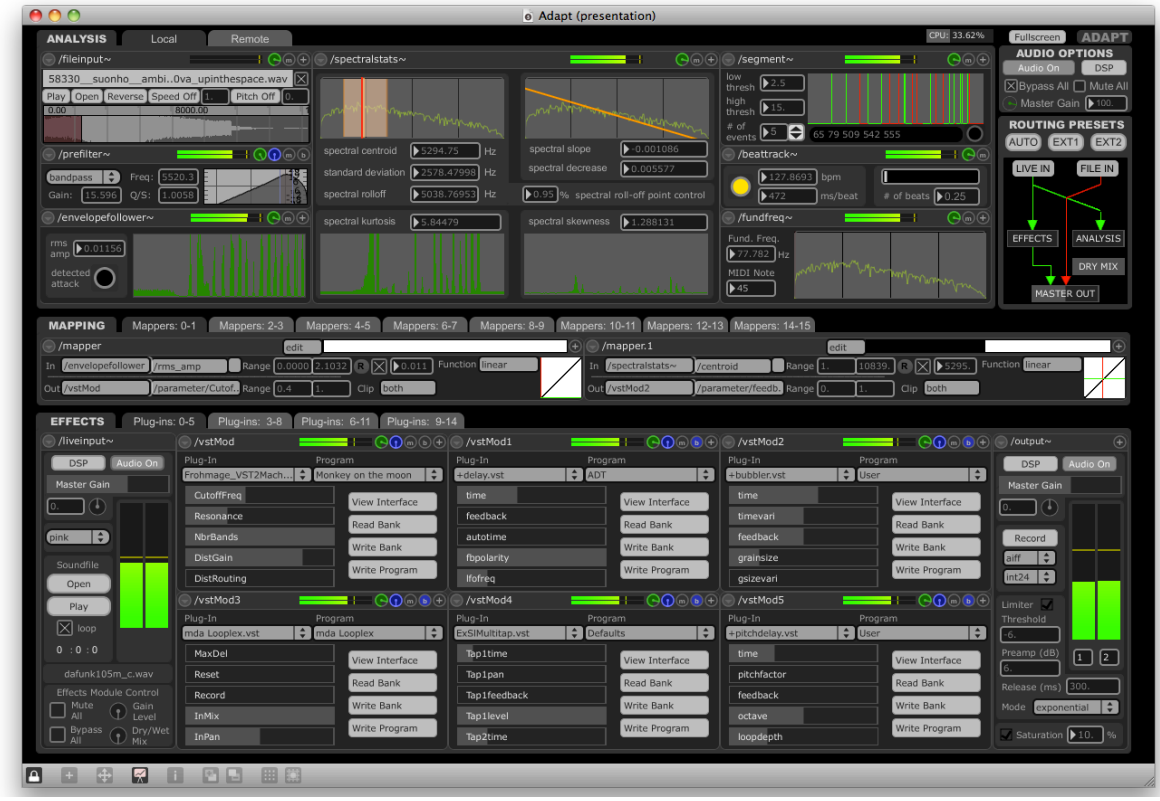

Figure 6: Screen shot of Adapt Interface

A. Audio Routing Options

Overview: The audio routing in Adapt defines the type of adaptive effect that may be created, which includes either an auto-adaptive effect or an external-adaptive effect. An auto-adaptive effect is created when the signal feeding the analysis bank is the same as the signal feeding the effects bank. An external-adaptive effect is created when the signal feeding the analysis bank is different from the signal feeding the effects bank. The signal path is configured via the Routing Presets panel; either by selecting one of the three preset buttons or by adjusting the four cross-faders located within the panel.

Adapt has two audio source modules available, which are /fileinput~ (first module in analysis bank) and /liveinput~ (first module in effects bank). Each audio source may be routed to feed either the bank they reside in only, the opposing bank only, or a mixture of the two. The output signal may be any mix of the "dry" signals directly from the two audio sources and the "wet" signal coming from the effects bank. After mixing, the master output signal is sent to the /output~ module which resides at the end of the effects bank.

A visual representation of the signal path is provided, with the green lines representing the /liveinput~ audio signal and the red lines representing the /fileinput~ audio signal. When the cross-faders are used for advanced signal routing, the mixture of the signals is depicted with standard color mixing, with a signal containing 50% of each input source appearing as solid yellow. When a signal is being attenuated, the corresponding line reduces in brightness and then disappears completely when the signal is muted.

Routing Presets

There are three pre-defined routing presets, which represent the most common audio configurations for the program.

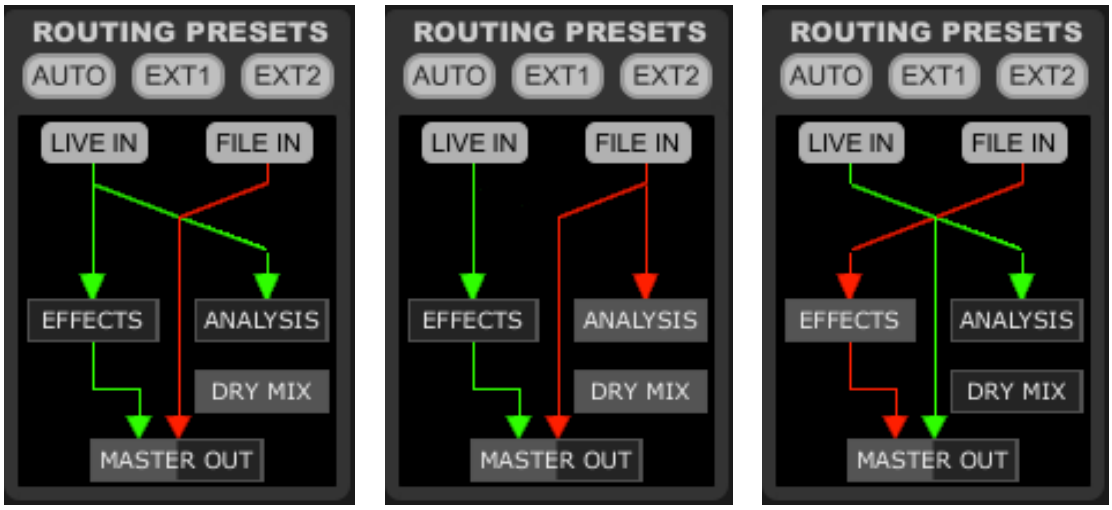

Figure 7: Routing Presets

1. **AUTO***:* Auto-Adaptive Effects Configuration

   The /liveinput~ source is routed both to the effects bank and the analysis bank, allowing for creation of auto-adaptive effects. The /fileinput~ dry signal is routed to the master output.

2. **EXT1:** External-Adaptive Effects Configuration 1

   The /liveinput~ source is routed to the effects bank. The /fileinput~ source is routed to the analysis bank, with the dry signal routed to the master output.

3. **EXT2:** External-Adaptive Effects Configuration 2

   The /fileinput~ source is routed to the effects bank. The /liveinput~ source is routed to the analysis bank, with the dry signal routed to the master output.

The above effects configurations define the possibilities available when only the "Local" tab of the Analysis Bank is in use. When networking is enabled and the "Remote" tab is used, an additional level of external-adaptive effects configuration is available to the user in addition to all of options available with the current local audio routing setup. Note that only remote feature extraction parameters are shared over the network, not the audio signal of the other performer.

Advanced Signal Routing

Advanced signal routing may be configured via the four cross-faders within the routing panel; they are labeled: "Effects", "Analysis", "Dry Mix" and "Master Out". Each slider has a range from 1.0 to 2.0, adjusting the slider performs an equal power fade between the two stereo signals connected to it, as defined below.

1. **"Effects" Cross-fader:** Controls the mix into the Effects bank, with 1.0 as 100% of the /liveinput~ sound source and 2.0 as 100% of the /fileinput~ sound source.
2. **"Analysis" Cross-fader:** Controls the mix into the Analysis bank, with 1.0 as 100% of the /liveinput~ sound source and 2.0 as 100% of the /fileinput~ sound source.
3. **"Dry Mix" Cross-fader:** Controls the mix of the dry signals into the master output, with 1.0 as 100% of the /liveinput~ dry signal and 2.0 as 100% of the /fileinput~ dry signal.
4. **"Master Out:" Cross-fader:** Controls the master output mix to be sent to the /output~ module, with 1.0 as 100% of the output of the Effects Bank and 2.0 as 100% of the output of the dry signal mix.

B. Analysis Bank

Overview: The analysis bank contains both time-domain and spectral-domain real-time feature extraction modules. Clicking on a module's name in the top bar of the module (i.e. /spectralstats~) displays a menu that lists all of the possible OSC parameters, messages, and return values for the module. The input signal to the analysis bank can be selected to be any mixture of the /fileinput~ module or the /liveinput~ module. Any signal input into the analysis bank is first sent through the /prefilter~ module and then sent to each of the analysis modules in parallel.

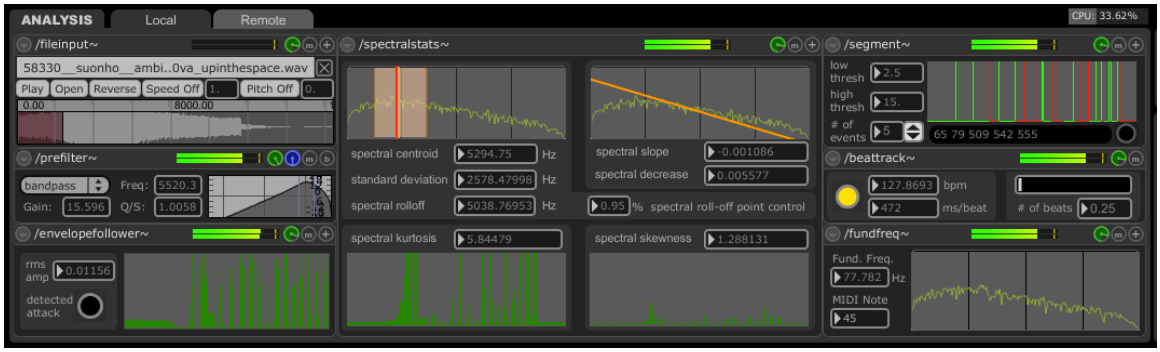

Figure 8: Analysis Bank

Utility modules in the analysis bank:

/fileinput~

- Fully featured .wav and .aiff sound file player. Although it is the first module in the analysis graph, the audio output may also be routed to the effects bank.

/prefilter~

- Stereo 2nd order IIR filter. Provides optional filtering before audio signal enters analysis bank.
- Filter Types: lowpass, highpass, bandpass, bandstop, peaknotch, lowshelf, highshelf

Analysis Module	Function	Extracted Features	OSC Return Address
/envelopefollower~	Extracts envelope of the input signal	RMS Amplitude	/rmsamp
/spectralstats~	Extracts various spectral signal descriptors from the input signal. Uses the Fast Fourier Transform.	Spectral Centroid Standard Deviation Spectral Rolloff Kurtosis Skewness Decrease Slope	/centroid /deviation /rolloff /kurtosis /skewness /decrease /slope
/segment~	Performs onset detection on the input signal.	Detected Event Inter-Onset Interval 1 Inter-Onset Interval 2 Inter-Onset Interval 3 Inter-Onset Interval 4 Inter-Onset Interval 5	/event /IOI1 /IOI2 /IOI3 /IOI4 /IOI5
/beattrack~	Estimates the tempo of the input signal.	Tempo in BPM	/bpm /msbeats
/fundfreq~	A fundamental frequency estimator.	Fund.Frequency (Hz) MIDI Note Value	/fundfreq /midinote

Table 1: List of all analysis modules

List of all analysis modules, extracted features, OSC return values and input parameters:

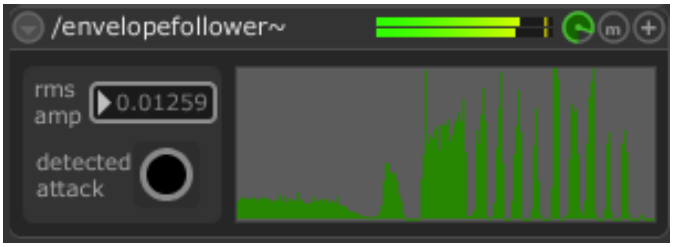

Figure 9: /envelopefollower~ module

/envelopefollower~:

- An RMS-based amplitude envelope follower, features a built-in low-pass filter and attack detector.

- Return value:
  - /rmsamp: current rms amplitude of the sampled window

- Input Parameters:

- /cutoff: lowpass filter cutoff frequency in Hz
- /period: sample size in ms
- /attack/threshold: sets the threshold for a detected attack
- /attack/interval: minimum number of ms between attacks

- Module created using the Max/MSP object "envelope-follower~" by Matthew Wright. Part of the CNMAT Max/MSP/Jitter Depot: http://cnmat.berkeley.edu/downloads

Figure 10: /spectralstats~ module

/spectralstats~:

- Computes seven MPEG-7 spectral descriptors in real-time using the Fast Fourier Transform. The FFT operation is shared for each descriptor computation, with a fixed FFT size of 2048 samples and a hop size of 4 (512 samples).

- Return Values:
  - /centroid: Spectral centroid, the barycentre of spectra
  - /deviation: Standard deviation of the spectral Spread, the variance around the specral centroid
  - /rolloff: Spectral roll-off point, the frequency so that x% of the signal falls below it.
  - /kurtosis: Spectral Kurtosis, a measure of the flatness of the spectrum around its barycentre.
  - /skewness: Spectral skewness, a measure of the asymmetry of a spectrum around its barycentre.
  - /slope: Spectral Slope, an estimation of the amount of spectral magnitude decreasing, computed by a linear regression of the magnitude spectrum.
  - /decrease: Spectral decrease, similar to the spectral slope, but a calculation supposed to be more correlated to human perception.
- Input Parameters:
  - /rolloff_point: sets the % roll-off point for the spectral roll-off descriptor.
  - /control_rate: sets the interval in ms to report the descriptors.

- Module created using Max/MSP objects from the Zsa.Descriptor library by Mikhail Malt and Emmanuel Jourdan. [3] http://www.e--j.com/?page_id=83

Figure 11: /segment~ module

/segment~:

- An event detection module that uses a filter bank to detect sharp changes in the spectral envelope of the input signal.

- Return values:
  - /event: Returns a 1 when an event is detected, 0 otherwise.
  - /IOI1: First computed inter-onset interval from first detected attack in a series of x attacks as specified by the input parameter /num_attacks
  - /IOI2: Second computed inter-onset interval, 0 if not computed
  - /IOI3: Third computed inter-onset interval, 0 if not computed
  - /IOI4: Fourth computed inter-onset interval, 0 if not computed
  - /IOI5: Fifth computed inter-onset interval, 0 if not computed
- Input Parameters:
  - /num_attacks: number of inter-onset intervals to group together and calculate
  - /low_thresh: low detection threshold
  - /high_thresh: high detection threshold

- Module created using the Max/MSP object "bonk~" by Miller Puckette, Ted Apel and Barry Threw. [14] http://crca.ucsd.edu/~tapel/software.html

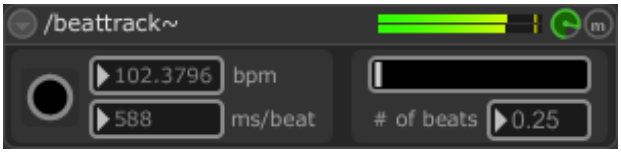

Figure 12: /beattrack~ module

/beattrack~:

- A FFT based beat-tracker, reports the estimated tempo of the signal in bpm every 1.5 seconds. [7]

- Return value:
  - /bpm: the estimated tempo of the signal in beats per minute.
  - /ms_beats: the number of ms/beat, calculation defined by /num_beats.

- Input Parameters:
  - /num_beats: The number of beats to use for the ms_beats calculation.

- Module created using the Max/MSP object "beattrack~" by Adam Stark. [8] http://www.elec.qmul.ac.uk/digitalmusic/beatfx/

Figure 13: /fundfreq~ module

/fundfreq~:

- Estimates the fundamental frequency of the input signal in Hz.

- Return value:
  - /fundfreq: fundamental frequency of the input signal in Hz.
  - /MIDI_Note: The estimated midi note value

- Module created using Max/MSP objects from the Zsa.Descriptor library by Mikhail Malt and Emmanuel Jourdan. [3] http://www.e--j.com/?page_id=83

Figure 14: Remote Analysis Bank

Remote Signal Analysis Parameters

The "Remote" analysis bank visually looks nearly identical to the "Local" Analysis bank with the exception of the first 2 utility modules, which are replaced by the UDP

network connection options. The analysis modules in the remote bank are actually just visual representations of the transmitted data from the other program's local analysis bank. Thus, the remote analysis modules do not include any analysis algorithms and only function to route incoming OSC data to the visual indicators and then broadcast the remote analysis parameters to the mapping bank.

C. Mapping Bank

Overview: Each module in the mapping bank allows for a continuous one-to-one mapping to be dynamically created between any control signal in the analysis bank and any effects parameter in the effects bank. A mapping is designated by first selecting an analysis module and analysis parameter using the drop-down menus labeled "In". The menus labeled "Out" are then used to define the destination effects module and parameter. Each drop-down menu is automatically populated with all modules and parameters within the program. A selection of signal conditioning operations may then be performed on the control signal. There are currently 16 individual mapping modules available, which can be selected via the tabs above the mapping bank.

Figure 15: Single Mapper Module

Signal Conditioning Parameters:

- Range: Minimum and maximum values of the input and output signals to be used for scaling. The small toggle button (with the "X") enables or disables automatic minimum and maximum range setting for the Input signal. The small "R" button resets the range to min = 1, max = 0 so that a new min. and max. may be automatically detected.
- Clip: Define clipping action to perform when lowest/highest range is exceeded. Possible Values are None, Low, High and Both.

- Function: Defines linear or non-linear transfer function for warping. Possible values are linear, logarithmic, exponential, lowpass, power, cosine and hyperbolic tangent.
- Ramp: Time in ms to ramp from one value to the next. (located on control panel)
- Slide: A logarithmic smoothing filter. Separate smoothing coefficient available for upward and downward movements. (located on control panel)

Mapper Module Edit modes:

- Bypass: The mapper is made inactive.
- Edit: Range and clipmode are set automatically according to the parameter/return settings; the transfer function may be edited.
- Active: Automatic range/clipmode setting, and curve display are disabled (to improve performance).

D. Effects Bank

Overview: The effects bank is an audio plug-in graph similar to those found in any VST host intended for live performance. The instrument signal is selected using the /liveinput~ module on the left side and enters the bank of VST plug-ins at the top-left slot. The signal flow through the bank of plug-in modules follows from left to right and then wraps back to the left when a new row is encountered. This mimics the signal flow of a typical guitar effects pedal board setup. There are 15 individual VST modules available and at the end of the graph the signal is mixed into the /output~ module as defined in Adapt's routing panel.

Figure 16: Effects Bank

VST Module Features:

- Each VST module automatically loads the drop-down menu labeled “Plug-In” with the names of all plug-ins residing in the standard VST directory on a Mac (/Library/Audio/Plug-Ins/VST/) or Windows (C:/Program Files/Steinberg/Vstplugins) machine.
- When a VST is selected, the module dynamically creates an OSC address for each parameter of the VST plug-in [15].
- The module also creates a horizontal slider for the first 5 parameters on the visible interface; any additional parameter sliders can be accessed by clicking on the (+) at the top-right corner of the module.
- Clicking the “View Interface” button will display the VST’s GUI (if it has one) or will generate one if not. The “Program” menu, “Read/Write Bank” and “Write Program” options are used for preset management.

V. Results

After a brief evaluation period, it is clear that the interface provided by Adapt indeed makes dynamic creation of new A-DAFx an extremely simple and intuitive process. As soon as the program is loaded, it is pre-configured for creation of auto-adaptive effects. Getting started is as simple as choosing a live input source or sound file, loading a VST plug-in, turning on the audio engine, and then configuring a mapping module by selecting a control signal source and effects parameter destination. The mapping bank’s auto-range function automatically detects minimum and maximum values in the control signal, which greatly improves the dynamics of the mapping and consistently yields musically pleasing results. Nothing more is required for creation of a

basic auto-adaptive effect, so the need for a quick proto-typing environment has been fulfilled. Fine-tuning the algorithm and creating complex mappings and routings is also very quick and easy; few restrictions are placed on the user in regards to configurability.

Probably the most unique and interesting feature is the ability to explore the potential of external-adaptive effects, both locally (with a secondary audio file) and remotely (with a live performer). For example, when sending a percussive track through the analysis bank and a melodic instrument through the effects bank, the results of the effects modulation are extremely pronounced and dynamic. If an analysis parameter such as the spectral centroid (which can be an indicator of "brightness" in the signal) is linked directly to a filter cutoff frequency of a resonant filter, the percussive hits can clearly be heard as timbral changes over the top of whatever melody is being performed. These types of modulations are incredibly easy to configure within Adapt, something that no other software can provide for a live performance situation.

Also, when experimenting with external-adaptive effects, once a mapping is assigned it is often interesting to hear the result if the two source signals are swapped, so that the one being modulated is now the one being analyzed and vice-versa. With Adapt, this can be done in real-time by simply clicking the correct preset button in the audio routing panel. The ability to switch back and forth quickly and even to cross-fade slowly between the different A-DAFx forms without interrupting the audio has proved to be an invaluable tool for exploring the many possible configurations.

VI. Future Work

A number of additions to Adapt will greatly increase its value for proto-typing A-DAFx algorithms. The most immediate need is the ability to globally save all module settings, so that once a complete adaptive effect is configured in the workspace, it may be easily recalled later. Achieving this would actually be quite simple, because each Jamoma module is in fact already capable of saving its own state to an XML preset file. Thus, a full session can technically already be saved by individually saving the state of each active model; creating a global preset system would follow quite easily.

Another addition of high value would be to make the analysis bank more easily extensible, much like the effects bank is able to load in any VST plug-in in any order. The current analysis bank setup has fixed Jamoma modules connected in parallel. Technically, anyone familiar enough with the Max/MSP environment could create additional analysis modules and add them to the bank by shuffling around the current modules. It would be much more ideal though to set-up the analysis bank so that new modules may simply be dropped into place and automatically connected to the sound source. Then, the desired analysis modules could be chosen from a menu and the user could configure the analysis bank as desired.

The mapping bank could also use some significant improvements, mainly along the same lines of enhanced extensibility. The mapping utilities currently provided are sufficient, but there are many more that could be borrowed from research involving gestural control of sound synthesis. The most elegant solution would be the addition of some kind of scripting capabilities, which allow the user to define new algorithms and equations for signal conditioning.

VII. Conclusion

Mapping adaptive control signals to the parameters of effect algorithms is a technique that can add interesting textures and dynamics to a musical performance or recorded piece. Since theoretically any analysis algorithm can be mapped to any effects parameter via a variety of mapping schemes, a multitude of possibilities exist for creation of complex and novel A-DAFx. A number of software programs allow for partial exploration and proto-typing of A-DAFx, but none exist that provide all of the necessary utilities and algorithms in a single package. Adapt provides a solution to this need by allowing dynamic creation of continuous mappings between a collection of extracted sound features and any parameter of any digital audio effect plug-in. The intuitive graphical user interface allows musicians without programming knowledge to explore creation of novel A-DAFx for the first time. At the same time, the complete extensibility of Adapt makes it a valuable research and proto-typing tool for advanced A-DAFx algorithm developers. The ability to share analysis parameters over a network allows musicians to explore creation of external-adaptive effects and the sensation of directly influencing another performers effects processing and musical output. These unique features of Adapt coupled with its extensibility make it an invaluable tool for A-DAFx research, live musical performance and music production.

References:

[1] Verfaille, V., Zolzer, U., Arfib, D., "Adaptive Digital Audio Effects (A-DAFx); A New Class of Sound Transformations", IEEE Trans. on Audio, Speech and Language Processing, Vol. 14, No. 5, pp. 1817-1831, September 2006

[2] G. Peeters, A large set of audio features for sound description (similarity and classification) in the CUIDADO project. Cuidado projet report, Institut de Recherche et de Coordination Acoustique Musique (IRCAM), 2004.

[3] M. Malt and E. Jourdan, "Zsa.Descriptors: a library for real-time descriptors analysis", in 5th Sound and Music Computing Conference, Berlin, Allemagne, 31th july to August 3rd, 2008, p. 134-137. See http://www.e--j.com/?page_id=83.

[4] D. Arfib, Recherches et Applications en Informatique Musicale. Paris, France: Hermès, 1998, ch. Des Courbes et des Sons, pp. 277–86.

[5] X. Amatriain, J. Bonada, A. Loscos, J. L. Arcos, and V. Verfaille, "Content-based transformations," J. New Music Res., vol. 32, no. 1, pp. 95–114, 2003.

[6] E. Perez_Gonzalez, and J. Reiss, "An automatic gain normalisation technique with applications to audio mixing," in *AES*, 2008.

[7] A. M. Stark, M. D. Plumbley and M. E. P. Davies. Audio effects for real-time performance using beat tracking. In Proceedings of the the 122nd AES Convention, Convention paper 7156. Vienna, Austria, May 5-8 2007.

[8] M. E. P. Davies and M. D. Plumbley. “Context- dependent beat tracking of musical audio”. IEEE Transactions on Audio, Speech and Language Pro- cessing, 15(3), 1009-1020, 2007.

[9] Malloch, J., Sinclair, S., and M. M. Wanderley (2007). From controller to sound: Tools for collaborative development of digital musical instruments. In Proceedings of the 2007 International Computer Music Conference, Copenhagen, Denmark, 2007, pp. 65-72.

[10] Malloch, J., Sinclair, S., and M. M. Wanderley (2008). A network-based framework for collaborative development and performance of digital musical instruments. In R. Kronland-Martinet, S. Ystad, and K. Jensen (Eds.): CMMR 2007, - Proc. of Computer Music Modeling and Retrieval 2007 Conference, LNCS 4969. Berlin Heidelberg: Springer-Verlag, pp. 401–425, 2008.

[11] Bencina, R. (2006) “Creative Software Development: Reflections on AudioMulch Practice,” Digital Creativity, 17(1), pp. 11 - 24, Routledge.

[12] T. Place and T. Lossius. Jamoma: A modular standard for structuring patches in max. In Proceedings of the International Computer Music Conference, pages 143–146, New Orleans, LA, 2006.

[13] T. Place, T. Lossius, A. R. Jensenius, N. Peters, and P. Baltazar. Addressing classes by differentiating values and properties in osc. In Proceedings of the 2008 Conference on New Interfaces for Musical Ex- pression, Genova, Italy, 2008.

[14] M. Puckette, T. Apel,. « Real-time audio analysis tools for Pd and MSP ». Proceedings, International Computer Music Conference. San Francisco: International Computer Music Association, 1998, pp. 109-112.

[15] M. Zbyszynski and A. Freed. Control of VST plug-ins using OSC. In Proceedings of the International Computer Music Conference, pages 263–266, Barcelona, 2005.

[16] V. Verfaille and D. Arfib, "Adafx: Adaptive digital audio effects," in *Proceedings of the COST-G6 Workshop on Dig- ital Audio Effects (DAFx-01), Limerick, Ireland*, December 2001.

[17] G. E. Krasner and S. T. Pope. A cookbook for using the model-view controller user interface paradigm in smalltalk-80. *J. Object Oriented Program.*, 1(3):26–49, 1988.