IMPLEMENTATION STRATEGIES FOR ADAPTIVE DIGITAL AUDIO EFFECTS

Verfaille V., Arfib D.

LMA - CNRS
31, chemin Joseph Aiguier
13402 Marseille Cedex 20
FRANCE
{verfaille, arfib}@lma.cnrs-mrs.fr

ABSTRACT

Adaptive digital audio effects require several implementations, according to the context. This paper brings out a general adaptive DAFx diagram, using one or two input sounds and gesture control of the mapping. Effects are classified according to the perceptive parameters that the effects modify. New adaptive effects are presented, such as martianization and vowel colorization. Some items are highlighted, such as specific problems of real-time and non real-time implementation, improvements with control curve scaling, and solutions to particular problems, like quantization methods for delay-line based effects. To illustrate, musical applications are pointed out.

1. INTRODUCTION

The use of digital audio effects has been developing for the last thirty years. They are extensively used for composition, mastering, real-time interaction, movies sound effects, etc. Several implementation techniques are been used, such as sample-by-sample, block-by-block treating, FIR and IIR filtering, delay lines, etc. New and adaptive effects may need new implementation schemes, even if they inherite from classical implementation schemes. In our case, introducing an automatic control level inside the effect adds complexity to the implementation. Moreover, implementation has to be though carefully, depending on wether the effect is a real-time or a differed-time effect. The adaptive step added to the real-time effect provides a higher interaction between the effect's control and the musician. Features scaling is needed to allow the performer to explore musical and gestural spaces in different ways.

2. ADAPTIVE DIGITAL AUDIO EFFECTS

Adaptive digital audio effects (ADAFx, [1]) are effects which control values vary in time according to features extracted from the sound and mapping laws ([2], [3] pp.476-8). A general diagram is given in Fig.1. A first input sound is used for feature extraction (low-level and higher level, perceptive parameters). The mapping between features extracted and effect control values includes non-linearities as well as linear combinations; it can be modified by gesture parameters. The effect is applied to a second input sound. When the two input sounds are identical, the effect is called auto-adaptive; otherwise, it is called cross-adaptive. The gesture control on the mapping allows a higher control level, since it permits a gesture mapping on the feature mapping.

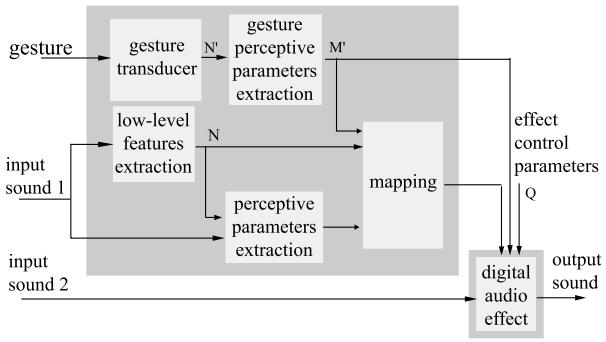

Figure 1: *Adaptive digital audio effects diagram: cross-ADAFx uses two different sounds whereas auto-ADAFx uses one sound. Gesture control is inserted in the mapping.*

2.1. Mapping functions

Let us consider K features, namely $\mathcal{F}_k(t),\ t = 1 \dots n_T$. Noting $\mathcal{F}_k^M = \max_{t\in[1;n_T]}(\mathcal{F}_k(t))$ and $\mathcal{F}_k^m = \min_{t\in[1;n_T]}(\mathcal{F}_k(t))$, an improved mapping function of the one proposed in [1] gives the expression of control curve $\mathcal{C}(t)$:

$$\mathcal{C}(t) = \Delta_m + (\Delta_M - \Delta_m)\ \mathcal{H}_{i_{cc}}(\mathcal{L}(t)) \quad (1)$$

$$\mathcal{L}(t) = \sum_{k=1}^{K} \frac{a_k}{\sum_{k=1}^{K} a_k} \mathcal{H}_{i_{cc}}\left(\frac{\mathcal{F}_k(t) - \mathcal{F}_k^m}{\mathcal{F}_k^M - \mathcal{F}_k^m}\right) \quad (2)$$

with a_k the weight of the k^{th} feature and the minimum Δ_m and maximum Δ_M values of the effect control parameter.

Let us now remind a few adaptive effects, processing perceptive parameters of sound [4].

2.2. Effects on the sound level

Effects such as compressor, noise gate, expander, limiter change the output level according to the input level: they perfectly illustrate the adaptive step in the mapping chain presented below. Knowing the input sound level, we modify the output sound level according to a non-linear law. But what happens when we change the output sound level using another sound feature than the input sound level, with a different law? For example, using the voiciness feature as input control and a mapping law such as $sin()$, we

obtain an effect that makes the vowels disappear and leaves only consonants.

2.3. Effect on time duration

Adaptive time-stretching [1], for instance using the phase vocoder processing [3], allows fine changes in time duration (a kind of re-interpretation of musical sentences) as well as strong changes (where the sound or musical sentence seems completely different).

To keep psycho-acoustic features such as vibrato, roughness, transition types, one should first analyse the sound. For example, in the case one wants to keep the vibrato aspect of the natural pitch shift produced by a singing voice or an instrument, it is necessary: first to extract the vibrato depth and rate [6] (for example using a likelihood model to detect sine waves [7]), second to apply a pitch-shift to erase the vibrato, third to apply the time-stretching, and fourth to apply a pitch-shift according to the vibrato parameters.

2.4. Effect on pitch

We used a Matlab implementation [3] of the pitch shifting using the Cepstrum technique [8]. We chose it because it is fast, since it uses Fast Fourier Transform implementation.

The **adaptive pitch-shift** is a simple pitch shift with the shift ratio given by a curve: the new pitch is then:

$$\mathcal{P}(t) = \mathcal{C}(t).\mathcal{H}_0(t) \tag{3}$$

with $\mathcal{H}_0$ the pitch of the original sound.

The **adaptive vibrato**, applied according to a harmonicity indicator value, is an automatic pitch modulation with a specific rate in $[4; 8]\ Hz$ and a specific depth in $[-\frac{1}{2}; \frac{1}{2}]$ tone, given by two control curves. It is important to pay attention to phase continuity of the modulation when the rate varies, otherwise the pitch may jump unpleasantly. Let $r(t_k)$ be the rate control curve and $d(t_k)$ the depth control curve of the vibrato. The resulting pitsh-shift ratio $\rho(t_k)$ is given by:

$$\rho(t_k) = 1 + \frac{d(t_k)}{2} \sin\left[2\pi\ r(t_k)\ t + \alpha(t_k)\right] \tag{4}$$

$$\mathcal{P}(t) = \rho(t).\mathcal{H}_0(t) \tag{5}$$

with $\alpha(t_{k+1}) = \alpha(t_k) + 2\pi\ t_k\ [r(t_k) - r(t_{k+1})]$. Moreover, a more complex adaptive vibrato can be designed using a transition type detector, in order to apply the vibrato only on stable parts of the sound [9].

A third example is the **pitch-change**. Using the control curve $\mathcal{C}(t)$ as a target pitch curve, the pitch-change is achieved by applying a pitch-shift with the following ratio:

$$\rho(t) = \frac{\mathcal{C}(t)}{\overline{\mathcal{H}_0}(t)} \tag{6}$$

where $\overline{\mathcal{H}_0}(t)$ is a corrected pitch curve for which zero values are replaced by the mean of non zero values of $\mathcal{H}_0$.

2.5. Effect on timbre

Adaptive filtering effects: when talking about "adaptive filtering", we first think about methods to estimate the parameters of a filter [10]. For example, in the field of telecommunications, adaptive filtering is used to minimize the feedback in a two channels communication system where the output of one channel (the phone) is near to the input of the second channel (the microphone). However, what we deal about is adaptive filtering effects, namely filters which properties (coefficients, bandwidth, formants, etc) evolve in time according to the mapping proposed. This is for musical purpose, and does not imply the use of the same techniques. We implemented adapted vocal-like filters, photosonic filters, wha filters (cf.[11] for the description of these filters), in real-time and it sounds great. In a way, it is a generalisation of the auto-wha, which is a wha-wha effect triggered by attack detection.

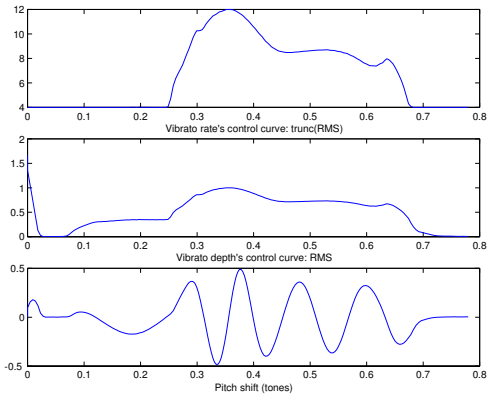

Figure 2: *Control curves for the adaptive vibrato: rate $r(t_k)$ (first figure), depth $d(t_k)$ (second figure) and the resulting pitsh-shift ratio $\gamma(t_k)$ (third figure).*

Robotization, whisperisation, granular adaptive delay are other known effects on timbre (already presented in a non-real time implementation [1]).

Martianization consists of an adaptive vibrato, with the rate and amplitude driven by continuous features outside of usual vibrato range. The rate r varies in $[0; 14]\ Hz$ and the amplitude around 1 octave instead of 1/2 tone. This effect gives wide variations in the pitch of a voice, loosing easily the sense of the message.

Vowel colorization or abusively called "vowel change" consists in recognizing the spectral shape of a vowel thanks to the cepstrum [8] of a Short-Time Fourier Transform, and replacing it by whitening the input signal and applying another spectral shape. The new spectral shape comes from reference vowel sounds, and the rules for changing are given by the user: combinatory rule, random rule.

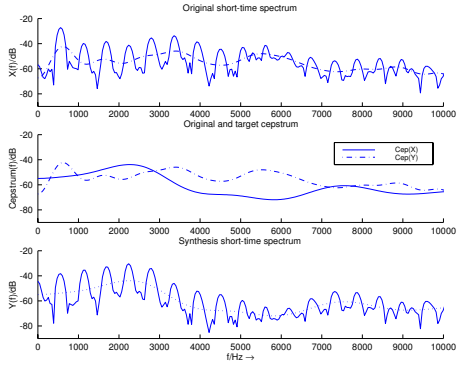

Figure 3: *Original STFT, original cepstrum, target cepstrum and synthesis STFT.*

Since vowel recognition and transformation is still a big challenge, we developped a simple and efficient enough vowel recognition scheme for musical transformations. It is still a work in progress: we will soon compare the efficiency of the proposed vowel recognition method with more complex and robust ones.

The recognition scheme is basically based on autocorrelation between reference cepstra and an analysis cepstrum. We compute the Short-Time Fourier Transform ($N_f = 2048$ points, with a 256 samples hop size) on a sliding Hanning window, and extract the spectral shape $\mathcal{S}(t)$ with the cepstrum technique (typically with a quefrency of 50). Let $v_i \in \{"a", "e", "i", "o", "u"\}$ be one of these five french vowels, and $\mathcal{S}^{ref}(v_i)$ be reference spectral shapes for the vowels. The vowel number is noted n_v.

For each reference spectral shape $\mathcal{S}(t)$ used to calibrate the algorithm, we compute the correlation:

$$\gamma(t, v_i) = \sum_{f=1}^{N_f} \mathcal{S}(t, f) \times \mathcal{S}^{ref}(v_i, f) \tag{7}$$

The calibrating sound is composed of the 5 vowels with known start and end frame numbers $[t^-_{v_i}; t^+_{v_i}]$, so we can compute the correlation means $\overline{\gamma}^{ref}(v_i)$, the normalized correlations $\overline{\gamma}^{ref}_{v_j}(v_i)$ and the normalized means $\gamma^{ref}_{v_j}(v_i)$:

$$\overline{\gamma}^{ref}(v_i) = < \gamma(t, v_i) >_{t \in [t^-_{v_i}; t^+_{v_i}]} \tag{8}$$

$$\gamma^{ref}_{v_j}(v_i) = \frac{\gamma^{ref}(t, v_i)}{\overline{\gamma}^{ref}(v_j)},\ t \in [t^-_{v_i}; t^+_{v_i}] \tag{9}$$

$$\overline{\gamma}^{ref}_{v_j}(v_i) = < \gamma^{ref}_{v_j}(v_i) >_{t \in [t^-_{v_i}; t^+_{v_i}]} \tag{10}$$

We define the distance $d^1_{v_j}(t, v_i)$ between an analyzed vowel and each reference vowel:

$$d_{v_j}(t, v_i) = \sqrt{\sum_{j=1}^{n_v} \left(\frac{\gamma_{v_j}(t, v_i) - \overline{\gamma}^{ref}_{v_j}(v_i)}{\overline{\gamma}^{ref}_{v_j}(v_i)} \right)^2} \tag{11}$$

The associated "quasi-probability" function $p_{v_j}(t, v_i)$ is the normalized inverse of the squared distance:

$$p_{v_j}(t, v_i) = \frac{(d_{v_j}(t, v_i))^{-2}}{\sum_{j=1}^{n_v} (d_{v_j}(t, v_i))^{-2}} \tag{12}$$

Being able to recognize which vowel is in the sound, we also need a *vowel detector*. The one we use is a threshold on the maximum value of $\gamma(t, v_i)$:

$$detec(t) = \max_i (\gamma(t, v_i)) > T_{detec} \tag{13}$$

This is a little bit rough, since the values of $detec$ are 0 or 1. The algorithm replace a spectral shape $\mathcal{S}(t)$ by the spectral shape of a vowel $\mathcal{S}_{v_i}$ is $detec(t) = 1$, and keep the original spectral shape otherwise (a consonant is not replaced or transformed). If we smooth the $detec$ curve, the transition is more slow. That way, in a differed-time algorithm, the changing starts during the end of the consonant, since the process is not causal (we know the whole detection curve before processing the colorization). However, for real-time implementation, we will have to anticipate the recognition and start the changes more rapidly. This work is in progress.

Rules for vowel color changing: when the vowel is detected and recognized, we can change it, for example according to a circular permutation, and apply a reference vowel to the whitened sound (or extracted source). The vowel colorization is achieved.

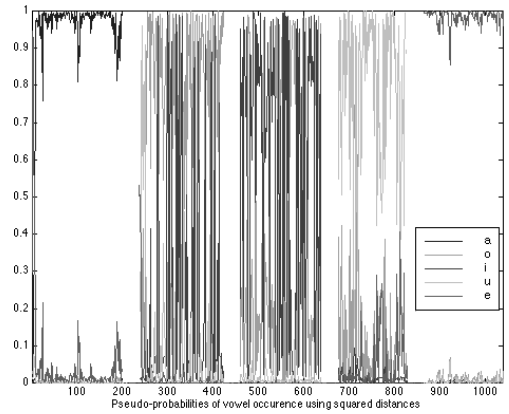

Figure 4: *Pseudo-probability of each vowel in the set $\{a, o, i, u, e\}$. Notice that the "a" and the "e" are clearly recongized, and that the three other ones not (they could be better recognized using a tracking method).*

Since our recognition scheme is not the best one, some errors appear. We prefer talking about vowel colorization instead of vowel changing at this stage of the project. Notice that to be efficient, this effect must use very clear reference vowels. They can be extracted from synthesis voice algorithms. We used the Voicer model [11] to produce synthesis vowels.

2.6. Effect on panoramisation

We first implemented a panoramisation driven by features. The left L_l and right level L_r are computed from control curves $\mathcal{C}_i(t)$. Using only one control curve, the effect is just an adaptive constant power panoramization (from Blumlein law [12], [3] pp.138-41), with $\theta = \mathcal{C}(t) \in [-\frac{\pi}{4}; \frac{\pi}{4}]$ and:

$$L_l = \frac{\sqrt{2}}{2}(\cos\theta + \sin\theta),\quad L_r = \frac{\sqrt{2}}{2}(\cos\theta - \sin\theta) \tag{14}$$

Another effect is obtained by assigning two different control curves to the channels:

$$L_l = \mathcal{C}(1, t),\quad L_r = \mathcal{C}(2, t) \tag{15}$$

This is no more a panoramisation, but the first step to spatialisation: indeed two independant levels for left and right channels gives left-right and front-back movements to the sound. Notice that the Interchannels Cross-Correlation (ICC) is not taken into account.

All the spatialisation perceptive parameters, such as the distance from the listener, elevation and azimuth, distance between the source and the room walls, etc, can be driven by sound features. A work in progress concerns the adaption of real-time spatialisation models (the Ircam Spatializer or the Gmem Holophon).

3. IMPLEMENTATION STRATEGIES

Several implementation strategies for the effect itself have been tested. First, we deal with the mapping stage of the process. We

then present the frame-by-frame implementation, the block-by-block implementation. Finally, we give solutions to the control curve quantization problem and functions for adaptive scaling of a control curve.

3.1. Mapping from sound features to control curve

The mapping explained in 2.1 has been implemented in Matlab and Max/MSP. The Matlab interface is a GUI with three features, that the user transforms to obtain a fixed mapping function $\mathcal{L}(t)$. The Max/MSP interface uses four features as input, and gesture control to change the weights a_k between the four transformed features, using an interpolator. The main difference is that the real-time version allows transformations of the mapping (for example going from a normal voice to a martian voice by moving a foot controller), whereas the non-real-time does not.

3.2. General frame-by-frame implementation

The frame-by-frame implementation scheme is the simplest one. Given constant frame lengths and frame hop sizes for analysis and synthesis, one input frame is transformed in an output frame, overlaped and then added to the output sound.

3.3. Block-by-block implementation

For non real-time processing using files, the frame-by-frame process is expensive, due to numerous memory accesses. In order to treat long sound files, we implemented (using Matlab) programs treating block-by-block (reading, transforming and writing) a raw formated sound file. The output format is the same: a raw file in 16 bits, with one or several channels stored as columns. Moreover, in a real-time process with small frames, the effect is usually applied block-by-block with an overlap and with constant frame length and frame hop size. However, for adaptive effects, this can be easily generalised to varying frame length and frame hop size treatments. We just have to care about the block overlap, that must be greater or equal to the maximum frame length.

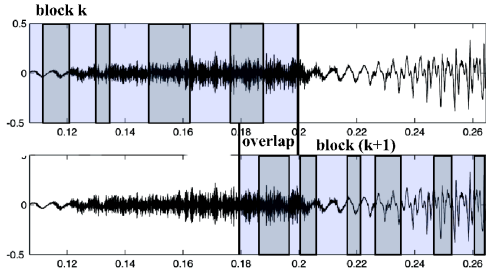

Figure 5: *Block by block implementation of effects with varying frame hop size and frame length. The buffer (or block) overlap must be greater or equal to the maximum frame length.*

Let us notice that the process itself is most of the time applied frame-by-frame, but data to process are given to the algorithm block-by-block. This means that there are two computation levels: one for the process itself (frame), one for the procedure (block).

This block-by-block implementation has been applied in differed time (using Matlab) for pitch-shift, pitch-change, vibrato, martianization, vowel colorization, robotization, whisperization, level change effect (all of them in their adaptive version). Robotization and level change effects have also been implemented in real-time (using Max/MSP). The other ones will soon be available.

3.4. Frame-by-frame implementation of time-stretching

Concerning the non-real-time adaptive time-stretching implementation, the input frame is read in irregular places, and the output frame is written in regular places. This gives a normalized output sound, but adds difficulties to the block-by-block implementation. Indeed, one may use bigger analysis blocks than the synthesis blocks, to permit big slowing downs. For example, to synthesize a N samples block speeding up T times the original sound, the analysis block length must be $N \times T$.

3.5. Frame-by-frame implementation of delay-line based effects

With delay line based effects, problems appear when any delay gain and/or delay time can be applied to a grain (for example with the adaptive delay). Hence, we cannot simply implement a block-by-block scheme without loosing precision on delay gain and delay time, and have to use a frame-by-frame scheme. This is really slow, due to the fact that output data are directly added in the output file by reading and overlapping the frames. The frame-by-frame non-real-time implementation algorithm is:

```
for k=1:nb_delay(k)
   p_out     = p_in + k*delay_time(k);
   fseek(fid_out, nOct*p_out, 'bof');
   DAFx_out = fread(fid_out, WLength, fOct);
   DAFx_out = DAFx_out + delay_gain(k) * frame_in;
   fseek(fid_out, nOct*p_out, 'bof');
   fwrite(fid_out, DAFx_out, fOct);
end
```

where `delay_gain` is an array with the possible values of delay gain, and `delay_time` is the array of delay times attributed to a value of the control curve.

It is not possible to implement it that way in real-time, since it would be necessary to have an infinity of delay lines. However, a block-by-block scheme can be implemented with a finite set of delay lines; then we need to quantize the control curve. This is explained in following subsection 3.6.

3.6. Feature quantization (needed for delay-lines based Fx)

When delay line effects are implemented as real-time processes (using Max/MSP) or block-by-block (in real-time and differed-time), the number of delay lines is limited. One way to best fit to the ideal sound (obtained with the non-real-time grain-by-grain implementation of granular delay) is to compute the quantization segments and values, and when a delay line is empty, to change its properties (length, gain) according to the most needed quantization value. That way, the delay line is re-allocated.

The control curves have to be quantized, according to a $n_q = 20$ or 30 values grid for example. The simplest solution is to use an **uniform grid** [13]. Let us consider a control curve $\mathcal{C}$ bounded by the interval $[\Delta_m; \Delta_M]$. The quantization segments:

$$\mathcal{I}(n, n_q) = [i_u(n, n_q); i_u(n+1, n_q)]$$

have $s_u(n, n_q) = \Delta_m + \frac{n-1/2}{n_q}(\Delta_M - \Delta_m)$ for middle values and $i_u(n, n_q) = \Delta_m + \frac{n-1}{n_q}(\Delta_M - \Delta_m)$ for bounds. The uniform quantified function is:

$$\mathcal{Q}_u\left(t, n_q\right) = s_u\left(\arg \min_{n \in \{1 \ldots n_q\}} |\mathcal{C}(t) - s_u(n, n_q)|, n_q\right) \quad (16)$$

Another solution consists in using a non-uniform quantization. A first one is the **non-uniform weighted quantization**. It consists in creating an histogram with $n_H > n_q$ points of the control curve and using the n_q greatest peaks as quantization values. The histogram function is $\mathcal{H}(n, n_H) = \sum_{k=1}^{N_T} \mathbb{1}_{\mathcal{C}(k) \in \mathcal{I}(n, n_H)}$ with the associated density function $\mathcal{D}(n, n_H) = s_u(n, n_H)$ and $\mathbb{1}$ the Heaveside function. The n_q maximum values of $\mathcal{H}(n, n_H)$ are $\delta(n), n = 1, .., n_q$ defined by:

$$\delta(n, n_q) = \mathcal{D}\left(\max_{k \in S(n-1)} \mathcal{H}(k, n_H), n_H\right)$$

with the set:

$$S(n-1) = \left\{ i \in \{1 \ldots n_H\}; \mathcal{H}(i, n_H) \in \{\delta(k, n_H)\}_{k=1 \ldots n-1} \right\}$$

That gives the weighted quantization function:

$$\mathcal{Q}_w\left(t, n_q\right) = \delta\left(\arg \min_{n \in \{1 \ldots n_q\}} |\mathcal{C}(t) - s_u(n, n_q)|, n_q\right) \quad (17)$$

This quantization does not take into account the local maxima and minima.

Musical sense can be given to a local peak value for the control curve according to the effect. That is the reason why we developed a second non-uniform quantization scheme taking into account local extrema called the **non uniform peak-weighted quantization**. After computing the $n_q - n_p$ weighted quantization values $\delta(n, n_q - n_p)$, we compute n_p quantization values corresponding to weighted values between the extrema of weighted quantization values and local extrema. Let us define the smallest interval containing all the weighted quantization marks:

$$\mathcal{I}_{extr} = [\min_n \delta(n, n_q - n_p); \max_n \delta(n, n_q - n_p)] = [\delta^-; \delta^+]$$

We extract $2n_p$ local maxima $\mathcal{P}^+(n)$ and minima $\mathcal{P}^-(n)$ and compute their distance to the nearest bound of $\mathcal{I}_{extr}$:

$$d^{\pm}(n) = \mathcal{P}^{\pm}(n) - \delta^{\pm}$$

We then define the weighted quantization marks:

$$\mathcal{P}_\alpha^{\pm}(n) = \delta^{\pm} + \alpha\left(\mathcal{P}^{\pm}(n) - \delta^{\pm}\right)$$

and their distance to the nearest bound of $\mathcal{I}_{extr}$:

$$d_\alpha^{\pm}(n) = \mathcal{P}_\alpha^{\pm}(n) - \delta^{\pm}$$

Finally, we classify them from the farest to the nearest to the interval bounds, and out of this interval:

$$\mathcal{P}_\alpha^{cl} = \left\{ \mathcal{P}_\alpha^{\pm}(k); \delta^{\pm} > 0, |\mathcal{P}_\alpha^{cl}(i-1) - m| > |\mathcal{P}_\alpha^{cl}(i) - m| \right\}$$

with $m = \frac{\delta^- + \delta^+}{2}$ the mid value of the $\mathcal{I}_{extr}$ interval. The set of quantization values becomes:

$$\overline{\Delta}(., n_q) = \{\delta(n, n_q - n_p)\}_{n \in \{1 \ldots n_q - n_p\}} \cup \left\{ \mathcal{P}_\alpha^{cl}(i) \right\}_{i \in \{1 \ldots n_p\}}$$

The peak-weighted quantization function we then obtain is:

$$\mathcal{Q}_{p,\alpha}\left(t, n_q\right) = \overline{\Delta}\left(\arg \min_{n \in \{1 \ldots n_q\}} |\mathcal{C}(t) - \delta(n, n_q)|, n_q\right) \quad (18)$$

For $\alpha = 0$, $\mathcal{Q}_{p,0}\left(t, n_q\right) = \mathcal{Q}_w\left(t, n_p\right)$: no peak is taken into account. For $\alpha = 1$, local extrema are directly taken into account. This means that near values to these peaks will be quantified to the peak value, which may produce a lower quantization error. Good values we used are $\alpha \in [0.5; 0.8]$.

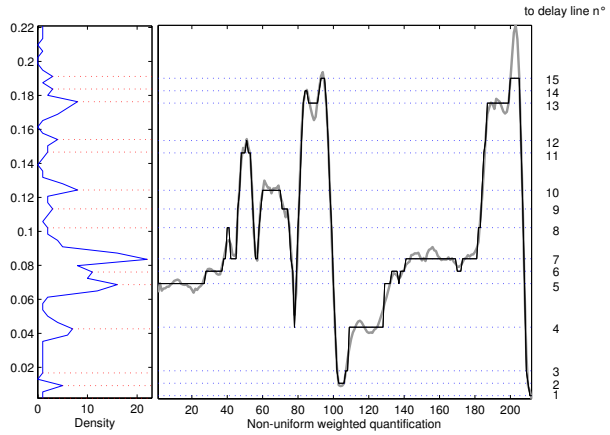

Figure 6: *Allocating a grain to a delay line by non uniform weighted quantization (right figure), using density function (left figure).*

The way to choose between one of the three quantization function (with several values for α) is not obvious. However, we can give a few clues. Firstly, for small grids (ie. $n_q \leq 30$), the granular delay with a quantized delay time control is clearly different from the frame-by-frame implementation. For example, comb filtering effects may appear. We recommend at least 60 delay lines concerning the delay time, and 20 concerning the delay gain. Secondly, the user may listen to the results of several quantization methods and control values: the musical effects can be very different, and there is no a priori feeling of how a quantization will sound good.

3.7. Control scaling

In the case of changing ranges curve, the effect can focalize in a zone of the control values when the sound parameter is confined in a small area, or when the user explores a small area with gesture transducer.

Let us consider the control curve $\mathcal{C}(t)$. We give several scaling (or zooming) functions, noted $\mathcal{Z}_i$, defined by the general formula:

$$\mathcal{Z}_i(t) = \mathcal{Z}_i(\mathcal{C}(t)) = \frac{\mathcal{C}(t) - \mathcal{Y}_i^-(t)}{\mathcal{Y}_i^+(t) - \mathcal{Y}_i^-(t)} \quad (19)$$

with the $\mathcal{Y}_i^+(t) = \mathcal{Y}_i^+(\mathcal{C}(t))$ and $\mathcal{Y}_i^-(t) = \mathcal{Y}_i^-(\mathcal{C}(t))$ functions defined as follows:

$$
\begin{aligned}
\mathcal{C}_T^-(t) &= \min_{k\in\{t-T\dots t\}} \mathcal{C}(k) \\
\mathcal{C}_T^+(t) &= \max_{k\in\{t-T\dots t\}} \mathcal{C}(k) \\
\mathcal{Y}_1^\pm(t) &= \mathcal{C}_T^\pm(t) \qquad (20)\\
<\mathcal{C}(t)>_T &= \frac{1}{T}\sum_{k=t-T}^{t} \mathcal{C}(k) \\
\mathcal{M}^+ &= \max, \quad \mathcal{M}^- = \min \\
\mathcal{Y}_2^\pm(t) &= \mathcal{M}^\pm\left(\mathcal{C}(t), \frac{\mathcal{C}_T^\pm(t) + <\mathcal{C}(t)>_T}{2}\right) \qquad (21)\\
\mathcal{Y}_3^\pm(t) &= \mathcal{M}^\pm\left(\mathcal{C}(t), \frac{\mathcal{C}_T^\pm(t) + \mathcal{C}(t-1)}{2}\right) \qquad (22)\\
\mathcal{D}_i^\pm(t) &= \delta^\pm \mathcal{C}(t)\ \left[|\mathcal{Y}_i^+(t-1) - \mathcal{Y}_i^+(t-2)|\right] \\
\mathcal{E}^\pm(t) &= \beta^\pm\left[1 - e^{\alpha\left(t-t_a^\pm\right)}\right] \\
\mathcal{G}_i^\pm(t) &= \gamma^\pm \mathcal{C}(t)\ \left[\mathcal{Y}_i^+(t-1) - \mathcal{Y}_i^-(t-1)\right] \\
\mathcal{Y}_4^\pm(t) &= \mathcal{M}^\pm\left(\mathcal{C}(t), \mathcal{C}_a^\pm + \mathcal{D}_4^\pm(t) \mp \mathcal{E}^\pm(t) + \mathcal{G}_4^\pm(t)\right) \\
\text{with} \quad \text{if} \quad & \mathcal{Y}_4^\pm(t) = \mathcal{C}(t), \text{then } \mathcal{C}_a^\pm = \mathcal{C}(t), t_a^\pm = t \qquad (23)
\end{aligned}
$$

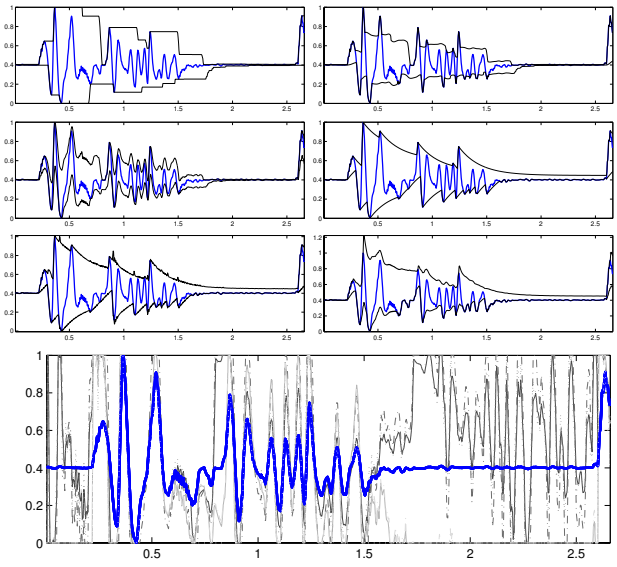

Figure 7: *Scaling in an area of the control parameter value with six different scaling functions. The most interesting is not the curves by themselves, but their different evolutions (six highest figures) and the different controls they provide (lowest figure).*

With $\mathcal{Y}_1^\pm(t)$, the bounds of the grid are given by filtering the local extrema values of the parameter, in a sliding frame. With $\mathcal{Y}_2^\pm(t)$, the filtered value and the mean value are taken into account. With $\mathcal{Y}_3^\pm(t)$, the filtered value and the last value are used. With $\mathcal{Y}_4^\pm(t)$, an exponential curve is used $\mathcal{E}$, as well as the derivative of the control curve $\mathcal{D}$ and the width of the last interval $\mathcal{G}$. According to the $\delta^\pm, \beta^\pm$ and $\gamma^\pm$ values, we can weight any of the three functions, and obtain really different control curves (cf. Fig.7).

4. CONCLUSIONS

ADAFx implementation requires different strategies according to the effect. The strategies can also differ for real-time implementation and non-real-time implementation. It is a large topic, with many orientations to be developed. Quantization is presented in the field of feature controls of delay-line based effects. Scaling feature is provided to controls, in order to be able to focus in a small range when the control parameter is confined in a small range. Musical sense can be given, transformed with adaptive effects. Real-time control is developing since it is useful for adding an expressivity level to the sound transformations.

5. REFERENCES

[1] Arfib, D., Verfaille, V., "A-DAFx: Adaptive Digital Audio Effects", *Proc. Workshop on Digital Audio Effects (DAFx-01)*, Limerick, Ireland, pp.10-4, 2001.

[2] Arfib D., "Des courbes et des sons", Recherches et applications en informatique musicale, Paris, Hermès, pp.277-86, 1998.

[3] *DAFX - Digital Audio Effects*, ed. Udo Zölzer, John Wiley & Sons, 2002.

[4] Amatriain X., Bonada J., Loscos A., Arcos J. L. and Verfaille V., "Addressing the content level in audio and music transformations", submitted for a special issue of the *Journal of New Music Research*, 2002.

[5] Portnoff, M.R., "Implementation of the Digital Phase Vocoder Using the Fast Fourier Transform", *IEEE Transactions on Acoustics, Speech and Signal Processing*, 24(3) pp.243-8, 1976.

[6] Arfib D., Delprat N., "Musical Transformations Using the Modification of Time-Frequency Images", *Computer Music Journal*, 17(2), pp.66-72, 1993.

[7] Verfaille, V., Charbit, M., Duhamel, P., "LiFT: Likelihood-Frequency-Time Analysis for Partial Tracking and Automatic Transcription of Music", *Proc. Workshop on Digital Audio Effects (DAFx-01)*, Limerick, Ireland, pp.82-6, 2001.

[8] Noll, A. M., "Short-time Spectrum and "Cepstrum" Techniques for Vocal Pitch Detection", *J. Acoust. Soc. Am.*, 36(2), pp.296-302, 1964.

[9] Rossignol S., Rodet X., Soumagne J., Collette J.-L. and Depalle P., "Feature Extraction and Temporal Segmentation of Acoustic Signals", *Proceedings of the ICMC*, ICMA, 1998.

[10] Haykin, S., *Adaptive Filter Theory*, Prentice Hall, Third Edition, 1996.

[11] Arfib, D., Couturier, J.M. , Kessous, L., "Gestural Strategies For Specific Filtering Processes", *Proc. Workshop on Digital Audio Effects (DAFx'02)*, Hamburg, Germany, 2002.

[12] Blauert, J., *Spatial Hearing: the Psychophysics of Human Sound Localization*, MIT Press, 1983.

[13] Zölzer, U., "Digital Audio Signal Processing", pp.19-21, ed. John Wiley & Sons, 1997.