A *Pure Data* toolkit for real-time synthesis of *ATS* spectral data

Oscar Pablo DI LISCIA
Universidad Nacional de Quilmes
Roque Saenz Peña 180
Quilmes, Argentina, 1876
odiliscia@unq.edu.ar

Abstract

This paper presents software development and research on the field of digital audio synthesis of spectral data using the *Pure Data environment* (Miller Puckette et al) and the *ATS* spectral analysis technique (by Juan Pampin [6]). The *ATS* technique produces spectral data using a deterministic-plus-stochastic representation. The focus is on the methods by which such data may be real-time read and synthesized using several *Pure Data* externals developed by the author and others, as well as on the involved audio synthesis strategies. All the software involved in this development are GNU Licensed and run under Linux.

Keywords

Digital Signal Processing, Sound Analysis, Computer Music.

1 The *ATS* analysis technique

1.1 General

The *ATS* technique (*Analysis-Transformation-Synthesis*) was developed by Juan Pampin. Its comprehensive study exceeds the goal of this paper[1] but, essentially, it may be said that it represents two aspects of the analyzed signal: the deterministic part and the stochastic or residual part. This model was initially conceived by Julius Orion Smith and Xavier Serra [11], but *ATS* refines certain aspects of it, such as the weighting of the spectral components on the basis of their *SMR*[2].

The deterministic part consists in sinusoidal trajectories with varying amplitude, frequency and phase. It is achieved by means of the depuration of the spectral data obtained using *STFT* (*Short-Time Fourier Transform*) analysis.

The stochastic part is also termed *residual*, because it is achieved by subtracting the deterministic signal from the original signal. For such purposes, the deterministic part is synthesized preserving the phase alignment of its components in the second step of the analysis. The residual part is represented with noise variable energy values along the 25 critical bands [12].

The ATS technique has the following advantages:

a-The splitting between deterministic and stochastic parts allows an independent treatment of two different qualitative aspects of an audio signal.

b-The representation of the deterministic part by means of sinusoidal trajectories improves the information and presents it on a way that is much closer to the way that musicians think of sound. Therefore, it allows many 'classical' spectral transformations (such as the suppression of partials or their frequency data transforming) in a more flexible and conceptually clearer way.

c-The representation of the residual part by means of noise values among the 25 critical bands simplifies the information and its further reconstruction. Namely, the common artifacts that arise in synthesis using oscillator banks or *IDFT*, when the time of a noisy signal analyzed using a FFT is warped may be completely suppressed[3].

1.2 Performing and storing ATS analysis

ATS was initially developed for the *CLM* environment (*Common Lisp Music* [5]), but at present there exist several *GNU* applications that can perform the *ATS* analysis, among them the *Csound* Package command-line utility *ATSANAL* [8][4], and the *ATSH* software (Di Liscia, Pampin,

[1] For a detailed reference of this technique, see [6].

[2] *Sound-to-Masking ratio*, see [12].

[3] This is possible because the residual part representation allows its synthesis using noise generators. More on this will be further explained in this paper.

[4] ATSANAL is based on the program ATSA (by Pampin, Di Liscia and Moss) and was ported to Csound

Moss [3]). The analysis parameters are somewhat numerous, and must be carefully tuned in order to obtain good results [2].

As documented in [3], the ATS files store a representation of a digital sound signal in terms of sinusoidal trajectories (called *partials*) with instantaneous frequency, amplitude, and phase changing along temporal frames. Each frame has a set of partials, each having (at least) amplitude and frequency values (phase information might be discarded from the analysis). Each frame might also contain noise information, modeled as time-varying energy in the 25 critical bands of the analysis residual.
The ATS files start with a header at which their description is stored (such as frame rate, duration, number of sinusoidal trajectories, etc.).
After the header data, the time, amplitude, frequency, phase and residual (these two may or may not be present) data of each partial in each frame are stored as 64 bits double values.
The format of the ATS files can be found in [3] but it is important to keep in mind that, at present, the ATS files come in four types:

Type 1: only sinusoidal trajectories with amplitude and frequency data on file.

Type 2: only sinusoidal trajectories with amplitude, frequency and phase data on file.

Type 3: sinusoidal trajectories with amplitude, and frequency data as well as residual data on file.

Type 4: sinusoidal trajectories with amplitude, frequency and phase data as well as residual data on file.
In Figures 2 and 3, plots of the Deterministic and of the Residual parts of a steady, 440 Hz sound of a Flute can be seen[5].

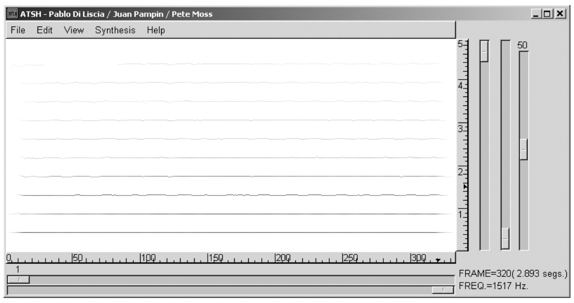
ATSH - Pablo Di Liscia / Juan Pampin / Pete Moss
File Edit View Synthesis Help
0 50 100 150 200 250 300
50
FRAME=320(2.893 segs.)
FREQ.=1517 Hz.

Figure 1: Plot of the Deterministic part of an *ATS* analysis.

by Itzvan Varga.

[5] These plots were obtained using the ATSH program (Di Liscia, Pampin and Moss) [3].

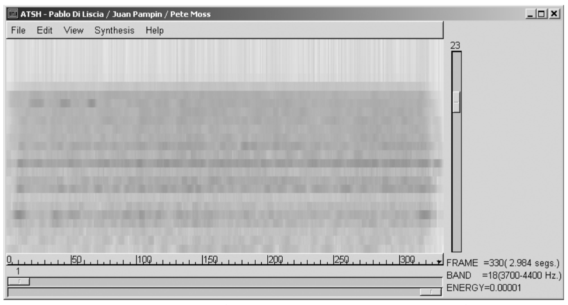

Figure 2: Plot of the Residual part of an *ATS* analysis.

2 *ATS* spectral data synthesis

At present, besides the original version in *CLM*, several *GNU* applications for *ATS* synthesis already exist. Several UGens for the *Csound* program were developed by Alex Norman [10]. Also, the *ATSH* program [3], allows the synthesis, editing and transformation of *ATS* data by means of a graphic interface. *SuperCollider* interfaces for *ATS* (including *classes* to read *ATS* files as well as UGens to do transformation and synthesis) are included in Josh Parmenter's UGen library, which is now part of the *SuperCollider* standard distribution.

2.1 *ATS* spectral data synthesis using *PD*

The synthesis procedure of an *ATS* analysis was designed in the following stages:
1-The *ATS* analysis data must be read from a file, parsed, and loaded in memory. The data of the file header (mainly the *ATS* file type, its duration, its frame rate, and the amount of partials per frame) must be also decoded and stored.
2-According to the *ATS* file type, the frequency, amplitude and phase (if any) and (if existing and required) the energy information of the residual part for each frame, must be sent at the right time (or with modifications that will warp the time of the original signal, if desired) to the synthesis units.
3-If the *ATS* file does not have residual data (types 1 or 2), using a bank with as many sinusoidal oscillators as partials, with their variable amplitude and frequency values for each frame properly interpolated would suffice to achieve the deterministic part synthesis.
4-If the *ATS* file has residual data (types 3 or 4), two special cases must be considered. The first case is when only the residual part is to be synthesized. The second case is when both, the deterministic and the residual part are to be synthesized.
In both cases, software units that produce random values in the range of -1 to 1 with periodicity *n*

(frequency $f = 1/n$) interpolating them linearly, are used as sources to reconstruct the original noise. As stated in [9][6], such units (which will be termed from here on *randi*[7]) produce a signal whose spectrum has a main lobe at 0 *Hz* and (*Nyquist_frequency*/*f* - *1*) lobes starting at ($f + f/2$), and spaced by frequency intervals of *f Hz*. The amplitude peak of the second lobe is approximately 24 *dB* below the amplitude peak of the main lobe, and the remaining lobes decrease even more in amplitude as their frequency rises. If such signal is multiplied by a sinusoidal signal of frequency *fc*, its main lobe will be centered in this frequency due to the convolution of the spectra of both signals, and the result is very similar to band-pass filtered noise with center frequency at *cf*. A plot of one of such signals is shown in Figure 3.

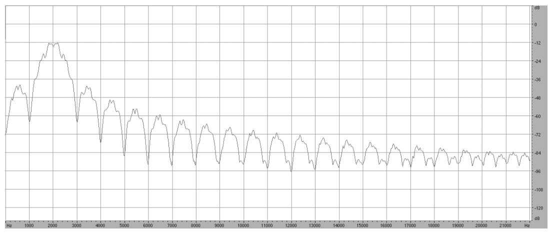

Figure 3: a plot of the spectrum of the output of a 1000 *Hz randi* unit convolved by a sine wave at 2000 *Hz* and sampled at 44.1 *KHz* A *FFT* of 1024 samples with a *Blackman-Harris* smoothing window was used in the analysis.

In the first case (only residual synthesis), the output signal is the sum of the output signals of 25 *randi* units with their frequencies set as the bandwidth of each one of the critical bands, each one modulated by a sinusoidal signal with its frequency set as the center frequency of the corresponding critical band. The amplitudes of each one of the resulting 25 signals are scaled by the *RMS* power values obtained in the residual analysis for each critical band and frame . The latter *RMS* values for each frame are linearly interpolated.

The second case is somewhat more complex. The residual energy values for each critical band must be ´re-injected´ into each deterministic sinusoidal trajectory according to the critical band where each one of these trajectories is located. Since this information is not stored in the *ATS* files, and to save computation time, this must be achieved before the real-time synthesis process. A function that evaluates the locations in the 25 critical bands of each frequency value for all the sinusoidal trajectories is used to compute and store in memory variable energy values of noise for each partial at each frame. The resulting signal for the synthesis of both, the deterministic part and the residual part of each partial, *k*, with frequency f_k, amplitude $a_k$and noise energy r_k, is made of the sum of the deterministic part (synthesized according to the procedure described in stage 3) plus the deterministic part (with changing frequency f_k, but scaled by the *RMS* values of noise for each partial, r_k, instead of the deterministic amplitude a_k), modulated by noise (produced by *randi* units with variable frequency proportional to f_k). The following formula shows the computing of a sample, *n*, of the output signal[8]:

$$output_n = \sum_{k=0}^{par-1} \left(a_k \sin(2\pi f_k n/R) + r_k \sin(2\pi f_k n/R) randi(f_k s) \right)$$

Where *par* is the number of partials (sinusoidal trajectories), *s* is a scaling factor and *R* is the Sampling Rate. The scaling factor *s* should be less than 1, because the resulting frequency values ($f_k s$) will determine the rate at which the *randi* units get a new random value and, essentially, the bandwidth of the resonant peaks of the output signal spectra[9].

The following sections will explain how several *Pure Data* externals were created and/or modified to properly connect them in order to perform the synthesis according to the stages and techniques above mentioned. It is assumed that the reader is familiar with the fundamentals of *Pure Data* programming, at the very least. The source code of all the externals, as well as some *ATS* files, and several *Pure Data* example patches can be found at [1].

2.2 Reading the data: The *atsread* external

The *atsread* external (by Alex Norman [7]), for *Pure Data*, was programmed to read *ATS* files and

[6] See pages 205-207.

[7] Just because the *randi* UGen of the *Csound* program generates this kind of signals.

[8] The deterministic part frequency and amplitude values (f_k, a_k) as well as the *RMS* power of the residual part values (r_k) must be obtained by interpolating the values of the actual frame and the next one according to the desired time.

[9] The bandwidth of the main lobe of the spectra will be of $2 f_k s$ *Hz*. A scaling factor of 0.1 (10% of f_k) seems to work fairly well for this application, but even a non- uniform scaling could be applied in order to get 'optimal' bandwidth values as a function of frequencies f_k.

to send its data to the required synthesis units. Essentially, this object takes an *ATS* file, parses its data and stores them in memory and allows sending them to the required synthesis units. The object expects a succession of *float*s indicating the time of the analysis that is required to be synthesized. According to the time value received, the frame of the analysis where it is located is fetched, and its corresponding data values are obtained by linear interpolation of the data of the current and the next frames, and sent through the respective *outlets*.
The author of this paper made several improvements to the *atsread* external in order to adhere to the needs of the synthesis units that are to be further used. The modifications are listed below and briefly explained, but their relevance will become more understandable in the next sections, where the synthesis units are explained.

1-Inclusion of an extra *outlet* for the output of the header data of the *ATS* file. As previously mentioned, knowledge of the type (whether residual data is present or not) and of the duration and number of partials of the opened file, at least, is needed in order to properly set the corresponding synthesis units and to send to them the time data. The header data is sent in the form of a *list* of *floats* once a file is opened.

2-Inclusion of a function to compute the residual noise values corresponding to each partial, for the case were residual information is present and both, deterministic and residual synthesis, are required[10]. If the file has residual data, then this function is called once it is opened, and the resulting values for each partial at each frame are stored in memory.

3-Inclusion of an extra *outlet* for the output of the noise values for each partial computed in 2.

4-Inclusion of an extra *outlet* for the output of the index (i.e., the partial number). The partial number if sent as a *float* value before its amplitude, frequency and noise (if any) data are sent.

5-Modification of the output format of the amplitude, frequency, phase and noise values. These are now sent as independent float values (with their partial index preceding them), instead of as *lists* of *float* values.

The modifications 4 and 5 were meant to simplify the connections with the synthesis units.

[10] Such *C* language function (*band_energy_to_res*) was taken from the analysis engine code of the program *ATSA* (by Pampin, Di Liscia and Moss).

2.3 Sinthesizing the deterministic part: The *oscbank~* external

If only the synthesis of the deterministic part is required, the *atsread* external may be connected with the *oscbank*~ external (by Richie Eakin, [4]). This external is designed to perform additive synthesis and produces its output audio signal through the use of a *lookup oscillator bank*. For better performance, the oscillator units are not interpolating, but as the default (sine) table length used is large enough (65536 values), the artifacts of truncation are minimized[11]. The *oscbank*~ external takes *control rate* successions of three *floats* (each one being respectively the oscillator number, its frequency and its amplitude). The received values are used to control the output of the corresponding oscillator and are interpolated linearly at a rate that can be set by the user. The default table and its size are appropriate for the purposes of the ATS deterministic part synthesis, but the user may experiment with other wave shapes and table sizes[12].

2.4 Sinthesizing the residual part: The *ats-noisy~* external

If only the synthesis of the residual part is required, the *atsread* external may be connected with the *ats-noisy*~ external (by Pablo Di Liscia). The output audio signal of this *external* is obtained adding the outputs of 25 *randi* units, each one with its frequency adjusted to the bandwidth of each critical band and modulated by a sine wave whose frequency is adjusted by the central frequency of each critical band. The output of each *randi* unit is scaled by the *RMS* power of the residual part of each critical band. These 25 *RMS* values for each analysis frame are received from the *atsread* external in the form of a *float list* and linearly interpolated. The interpolation rate may be set by the user.

2.5 Sinthesizing the deterministic and the residual parts: The *ats-sinnoi~* external

If the synthesis of both, the residual and the deterministic parts is required, the *atsread* external may be connected with the *ats-sinnoi*~ external (by Richie Eakin and Pablo Di Liscia).

[11] As stated by Moore ([9], pp. 166), a truncating oscillator reading a table of 65536 values, produces a *signal-to-error-noise ratio* of approximately 85 *dB*.

[12] Possibly obtaining strange, but musically interesting, results!

This external was programmed modifying the *oscbank~* external (by Richie Eakin).
Basically, the above mentioned modifications include the addition of a bank with as many *randi* units as deterministic partials are to be synthesized, plus an *inlet* to retrieve the residual RMS power data for each partial, which must be sent by *atsread* as a succession of *float* values. As explained in section 2.1, in this case the corresponding noise of the residual part must be re-injected in each deterministic trajectory. This is computed previously by the *atsread* external once a file having both residual and deterministic part is loaded. The residual part, in this case, is synthesized using *randi* units as sources, but the output of each one is multiplied by the output of each deterministic trajectory and scaled by the RMS of the noise computed for each partial. The deterministic part is synthesized as explained in section 2.3. An extra *audio outlet* was added to the external as well, in order to have individual outputs of both, the residual and the deterministic parts. This allows the user to further mix them with individual amplitude scaling, if desired. The user may synthesize only the deterministic part with this external as well, but doing so with the *oscbank~* external (as explained in section 2.3) will be much less CPU consuming.

3 Conclusions

The synthesized signals of all the presented units were found of similar quality of the ones produced by the synthesis units listed in the beginning of Section 2. Other strategies for synthesizing the residual part such as, for instance, a bank of *band-pass* filters processing *white noise* sources, could be used as well. However, the method described in this paper seems to provide a better 'blending' of the residual and deterministic signals with the additional benefit of being less CPU consuming.
The example Patches that were developed are simple, and only deal with the synthesis of all the partials with modifications in the duration of the output signal and its frequency. However, the examples suggest that, using the *Pure Data* toolkit that was presented in this paper, the user skilled in *Pure Data* programming may achieve very interesting transformations in a relatively straight forward way.
Future research will be focused in the development of an analysis external to perform high level analysis of the *ATS* data, with the purpose of allowing real time transformations of timbre and spectral morphing between different sounds.

4 Acknowledgements

The author thanks the *Universidad Nacional de Quilmes* (*UNQ*, Buenos Aires, Argentina) for supporting and hosting this research, to Juan Pampin (the developer of the *ATS* technique) for his advice, and to Alex Norman and Richie Eakin, the authors of the *Pure Data* externals that were taken as the basis for this development.

References

[1] Oscar Pablo Di Liscia. 2013. *PD-ATS Toolkit*. https://puredata.info/Members/pdiliscia/ats-pd

[2] Oscar Pablo Di Liscia and Juan Pampin. 2002: *ATSH Manual*, http://musica.unq.edu.ar/personales/odiliscia/software/ATSH-doc.htm

[3] Oscar Pablo Di Liscia and Juan Pampin. 2003. Spectral analysis based synthesis and transformation of digital sound: the ATSH program. *Proceedings of the IX Brazilian Symposium of Computer Music*, NUCOM, Minas Gerais, Brasil.

[4] Richie Eakin. 2007. *oscbank~*. https://github.com/pdl2ork/pd/tree/master/externals/oscbank~

[5] Juan Pampin. 1999. ATS: a Lisp environment for Spectral Modeling. *Proceedings of the International Computer Music Conference*, Beijin.

[6] Juan Pampin. 2011. *ATS_theory*, http://wiki.dxarts.washington.edu/groups/general/wiki/39f07/attachments/55bd6/ATS_theory.pdf

[7] Juan Pampin, Oscar Pablo Di Liscia, Pete Moss and Alex Norman. 2004. ATS user Interfaces. *Proceedings of the International Computer Music Conference*, Miami University, USA.

[8] Itzvan Varga, *ATSANAL documentation*, http://www.csounds.com/manual/html/UtilityAtsa.html

[9] F. Richard Moore.1990. *Elements of Computer Music.* Prentice-Hall., New Jersey, USA.

[10] Alex Norman. 2004. *Csound ATS spectral processing UGens*,

http://www.csounds.com/manual/html/SpectralAT S.html

[11] Xavier Serra and Julius O. Smith III. 1990. A Sound Analysis/Synthesis System Based on a Deterministic plus Stochastic Decomposition, *Computer Music Journal*, Vol.14 #4, MIT Press, USA.

[12] Ernst Zwiker and Hugo Fastl. 1990. *Psychoacoustics Facts and Models*. Springer, Berlin, Heidelberg.