PVLAB: an audio processing program based on Phase Vocoder

Remigio Coco

Classe di Musica Elettronica
Conservatorio "O. Respighi" - Latina
rcoco@jumpy.it

ABSTRACT

The Phase Vocoder is a well-known technique of signal analysis, successfully used for the analysis and resynthesis of audio signals. In this paper, a GUI program for audio signal processing, based on the Phase Vocoder, is presented. The name of the program is PVLAB (Phase Vocoder LABoratory), it is written in C++, and it is available only on Windows platform, for the moment. It can do several types of transformation, grouped into four categories: time transformations, frequency transformations, amplitude transformations and miscellaneous transformations (those transformations that involve more than one domain). It works on mono audio files in WAV format, or analysis files obtained with Csound; stereo files will be supported in future.

1. INTRODUCTION

Phase Vocoder analysis is carried out by means of short-term FFTs; input signal segments can be contiguous or overlapping, giving more or less time resolution. The result of the analysis phase is a number of *frames*, each containing (N/2 + 1) pairs {A,f} (amplitude, frequency), where N is the number of FFT points. PVLAB gives the opportunity to the user to set the number of FFT points, window type (Hamming, Bartlett, etc.) and overlapping amount (number of points between two consecutive frames of analysis – limited to powers of two). In PVLAB, the analysis data are processed further, as we will see, yielding a new set of analysis frames, and then the signal is resynthesized. The resynthesis technique is additive, considering a bank of (N/2 + 1) sinusoidal oscillators updated at each frame with the pairs {A,f} (ref. [1], [2], [3], [4], [5]).

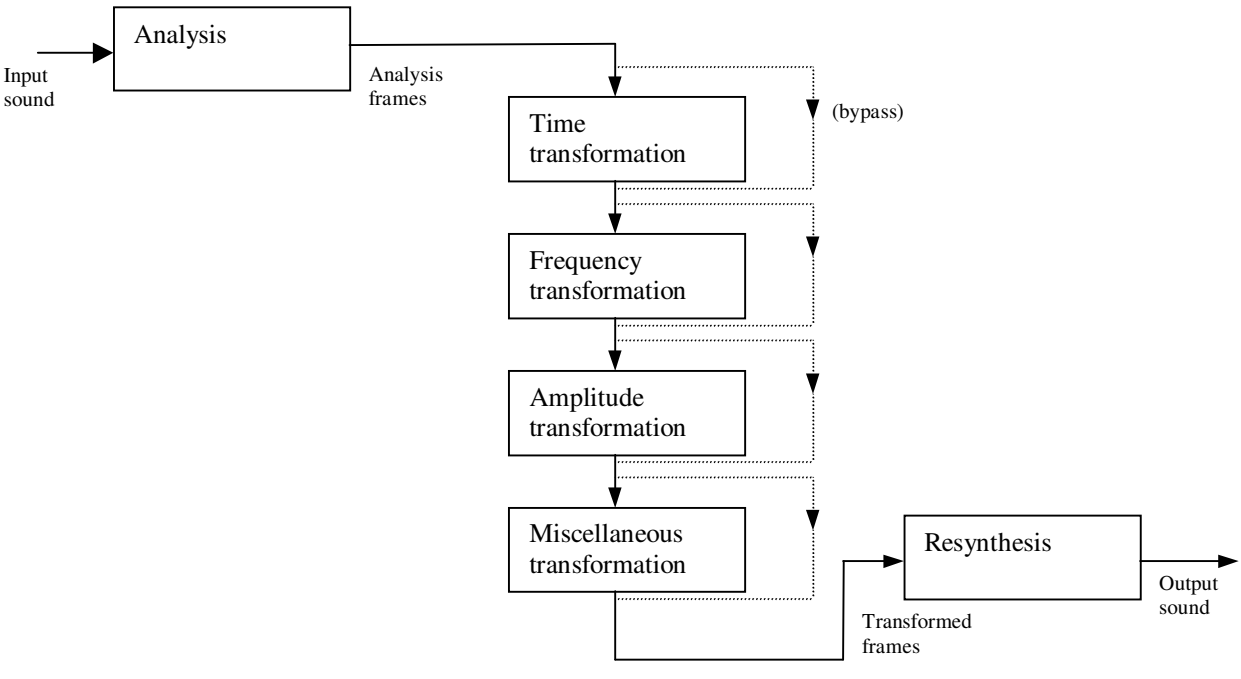

Fig. 1 - Block diagram of signal processing in PVLAB

1.1. Time transformations

In the time domain, there is only one transformation in PVLAB: the modification of the order in which the frames are written to the output. The user can define a curve {input time vs. output time}, also defining the output duration. The simplest application of this transformation is time contraction or expansion of the input signal: the user should choose a linear curve y=x and modify the output duration. Interpolation between adjacent frames can be enabled or disabled: if disabled, output frames are identical to input frames, except for the time location.

1.2. Frequency transformations

There are three transformations in PVLAB, belonging to the frequency group: *Remapping*, *Quantize* and *Random add*. *Remapping* is straightforward: the user defines a curve {output frequency vs. input frequency}, and each frequency value contained in the pair {A, f} is modified accordingly. The simplest effect achievable with this technique is pitch shifting: one has simply to define a curve of the form y=ax. PVLAB allow also the user to define two frequency-remapping curves and an interpolation curve, so that the spectral deformation can be varied in time. The *Quantize* transformation is a particular case of remapping: the frequencies of the output can assume only discrete values, calculated with the formula

$$f_n = f_0\, r^n \quad \text{with } n = \ldots -2,-1,0,1,2,\ldots \qquad (1)$$

where f_0 is the base frequency and r is the quantization ratio.

For example, if base frequency is 27.5 Hz and quantization ratio is 1.059463, the quantization grid corresponds to the frequencies of the temperated scale. For each {A, f} pair, the new frequency value equals the grid value nearest to the old value. Finally, in the *Random add* transformation, a random value is added to the original frequency value: the maximum value added is 4 times the frequency resolution, given by (sampling rate)/(n. of FFT points). The user can set the amount of randomness on a frequency basis, defining a curve {random amount vs. frequency}, or two curves and an interpolation curve.

1.3. Amplitude transformations

The amplitude transformations group contains three items: *Transfer function*, *Threshold*, *Inverse threshold*. The *Transfer function* is the classical curve {amplitude coefficient vs. frequency}, that allow the user to perform very precise filtering. Filtering can also be dynamic, interpolating between two filtering curves. *Threshold* and *Inverse threshold* operations are trivial: the output values of the amplitude are equal to the input values if these are above (or below, respectively) the threshold value, otherwise they are set to zero. The *Threshold* transformation can be used for noise reduction, or to extract the strongest harmonics of an instrumental sound, for example. In PVLAB, the threshold value is specified as a fraction (from 0.0 to 1.0) of the maximum RMS value of the input signal.

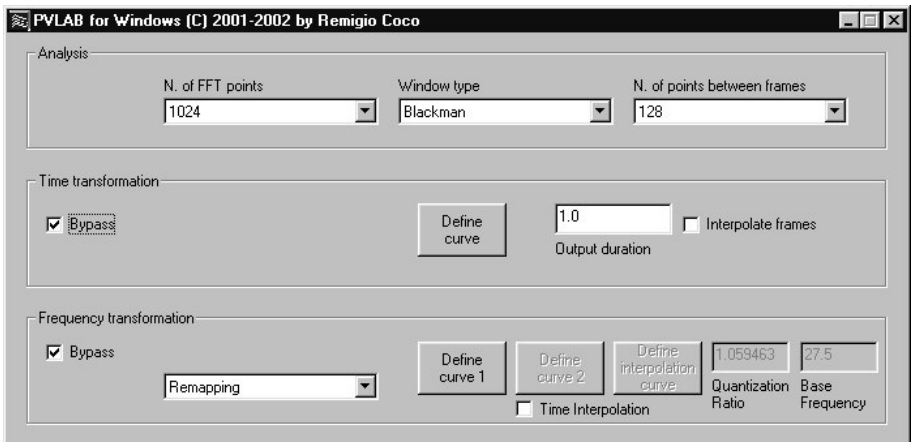

Fig. 2 Screenshot of the upper part of the main window

1.4. Miscellaneous transformations

This group of transformations include the following: *Delay vs. Frequency*, *Stereo-ization*, *Morphing (interpolation)*, *Morphing (cross-synthesis)*, *Morphing (density)*. In the first tranformation, the user can define a curve {delay vs. frequency}, so that each pair {A, f} of the output signal is delayed in comparison with the corresponding input pair; the amount of delay varies with the frequency (hence the name of the transformation). The user has to define the normalized {delay vs. frequency} curve (or two curves and an interpolation curve), and also the maximum delay (in seconds). The effect of this transformation is a sort of "spectral arpeggio", in which the partials of the input signal are not synchronized anymore, but they are presented in a different order. The *Stereo-ization* transformation generates a stereo output signal, based on a curve {Left-Right balance vs. frequency}. The amount of signal passed to each stereo channel varies with the frequency, so that the user could split, for example, high frequencies to the left, and low frequencies to the right. By defining two curves and an interpolation curve, one can also create interesting "moving" effects.

The three following transformations are similar, and are all sound morphing techniques. In all these, the user has to define two curves, {amplitude morphing factor vs. time} and {frequency morphing factor vs. time}; of course, he has to supply two audio or analysis files to the software, let's call them A sound and B sound.

The output signal duration is the maximum duration between A and B; the shorter sound is prolonged, repeating the last frame. In the *Morphing (interpolation)* technique, each output value of amplitude and frequency is calculated by linear interpolation between A values and B values, according to the morphing factor defined by the user (0.0 means only A, 1.0 means only B). In the *Morphing (cross-synthesis)* technique, each output value of frequency is obtained by linear interpolation, exactly as in the previous case. Instead, the output amplitude values are obtained "filtering" the A signal with the spectrum of B signal, and the morphing factor is simply a mixing factor between the original signal A and the filtered signal A#B. The cross-synthesis is achieved normalizing the amplitude values of B, dividing each value by the maximum RMS; then, for each frame and each frequency bin, the amplitude values of A are multiplied for the corresponding normalized value of B. Finally, in the *Morphing (density)* transformation, the morphing factor controls how many values of amplitude and frequency (separately) are taken from A and from B. For example, if amplitude morphing factor is 0.4, each frame is made of 40% amplitude values taken from B, and the remaining 60% from A; distribution of the values inside each frame is casual (i.e. the location of A and B values among the N/2+1 values is randomly chosen).

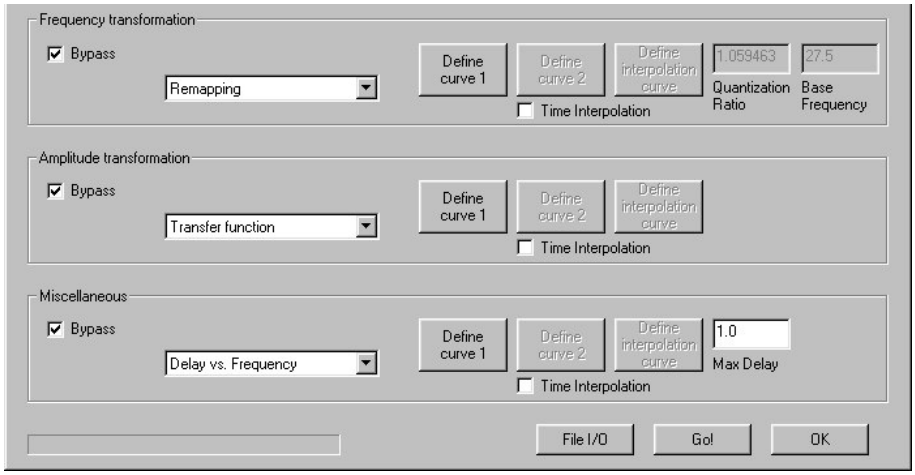

Fig. 3 Screenshot of the lower part of the main window

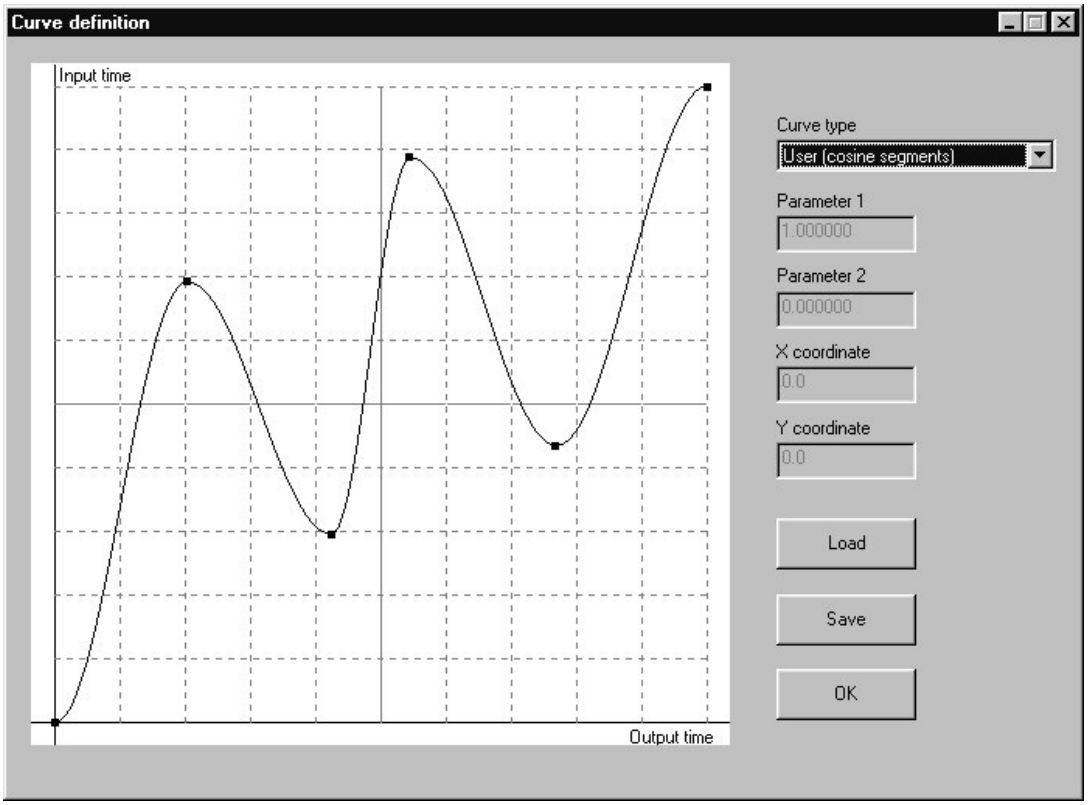

Fig. 4 - Screenshot of the curve definition window

2. CONCLUSIONS

The PVLAB program is useful in a huge range of audio processing needs. Its main features are speed of execution, in comparison with general purpose software like Csound, and the capability to define curves with an intuitive and easy-to-use GUI.

3. REFERENCES

[1] Dodge, C. and Jerse, T.A., "Computer Music. Synthesis, Composition and Performance", Schirmer Books, New York, 1985.

[2] Dolson, M., "The phase vocoder: a tutorial", Computer Music Journal Vol. 10, N. 4: pp. 14-27, 1986.

[3] Wishart, T., "The composition of Vox-5", Computer Music Journal Vol. 12, N. 4: pp. 21-27, 1988.

[4] Dobson, R.,"The operation of phase vocoder. A non-mathematical introduction to the Fast Fourier Transform", Composers' Desktop Project (CDP), York, 1993.

[5] Oppenheim, A.V., and Schafer, R.W., "Digital Signal Processing", Prentice Hall Inc., New Jersey, 1988.