SVP : A MODULAR SYSTEM FOR ANALYSIS, PROCESSING AND SYNTHESIS OF SOUND SIGNALS

Ph. Depalle & G. Poiret
IRCAM, 31 Rue Saint Merri, 75004 PARIS (FRANCE)
Tel: 13-142-22-12-33 Ext: 48-14. email: phd@ircam.ircam.fr, gfp@ircam.ircam.fr

ABSTRACT: Based on the design of a phase vocoder [9], we have developed SVP, a modular system for analysis, processing and synthesis of sound signals. SVP allows for great flexibility and covers a wide range of musical applications in a uniform manner. SVP is written in C using the Unified Dip Interface (UDI) programming environment [2] and may run on any array processor which supports UDI. SVP has two signal inputs and one signal output. Each input signal may be analyzed, or filtered on successive sliding windows. The two resulting spectral representations may be combined. Afterwards, the result is transformed into a time-domain signal. During this transformation, the time-scale can be modified in the classical way. Each operation can be dynamically controlled in time. SVP may be connected to any control parameter generator, for instance FORMES [1]. We also present a multichannel, open version of SVP called Extended SVP which allows the user to design its own algorithms for combating input signals.

I) INTRODUCTION

Based on the design of a phase vocoder, we have developed a modular system for analysis, processing and synthesis of sound signals, which is called SVP (for the acronym of the French name: Super Vocoder de Phrase).

The structure of SVP, which can be used as a standard phase-vocoder [8] as well as a sophisticated source-filter synthesizer [10], allows for great flexibility and covers a wide range of musical applications in a uniform manner. SVP is written in C using the Unified Dip Interface (UDI) programming environment [2] and may also run on any array processor which supports UDI. It is based on the overlap-add Short Time Fourier Transform (STFT) Analysis-Synthesis method, but also offers several spectral envelope estimators. SVP has two signal inputs and one signal output. Each input signal may be analyzed on successive sliding windows to produce a spectral representation (a STFT or a spectral envelope), and may be filtered by a transfer function multiplication. The two resulting spectrum representations may be combined for mixing, cross-synthesis, or ring-modulation. Afterwards, the result is transformed into a time-domain signal. During this transformation, the time-scale can be modified in the classical way. Each operation mentioned above (displacement of the sliding window, mixing, filtering, and time-scaling) can be dynamically controlled in time. SVP may be connected to any control parameter generator, for instance FORMES [1].

Section II of this paper will provide a description of the structure of SVP focusing on the most innovative aspects of the program, while section III will describe a multichannel and open version of SVP called Extended SVP.

II) SVP, A MODULAR SYSTEM

II.1. Introduction

The design of SVP is based loosely on an "object-like" approach which results in a modular program structure [see figure I]. At the moment, SVP is built on five types of modules. The Channel Module parses the input sound stream into consecutive, possibly overlapping blocks. The Analysis Module transforms each block to form a spectral representation. The Filtering Module processes the frequency data. The Combination Module combines two inputs to generate a single output. The Synthesis Module reconstructs the sound from the spectral data.
A module generates a single output from one or two data inputs and several control parameter inputs. Each module hides all of the specificities of its attached functionality. This simplifies the structure, maintenance and extensibility of the program and makes it easier to learn to use.

The Channel, Analysis and Filter modules make up a voice structure. SVP processes up to two voices with multichannel capability provided by an extended version of the program (See Sect. II).

II.2 Channel Module

The Channel Module parses the input sound file into blocks. A number of options provide the ability to move through the sound in non-uniform ways. These are grouped into two classes:

1) variable displacement allows the beginning of every temporal window to be specified individually. One can synchronize the analysis with localized temporal events such as percussive attacks. One can also move as desired through the sound, moving forwards, backwards, looping, spacing out or closing up the analysis windows or "freezing" the sound for a period of time.

2) synchronous displacement allows specification of a time varying sequence of "frequencies" whose periods are calculated to control the time increment through the sound. A simple example of the use of this option is pitch synchronous processing but other uses are also possible (twice or half the period length, etc.).

The size of the temporal windows can also be specified in a flexible manner. A variable size allows the analysis to adapt to the frequency content of the sound and to its temporal variability by optimizing the compromise between temporal and frequency resolution. This allows the use of large windows when required (e.g. presence of low frequency components in the signal) while avoiding smoothing of transients.

The window size can also be specified in a synchronous manner where the size is proportional to a frequency period whose temporal evolution is given in a file. A simple example is the use of the fundamental frequency of the sound to restrict analysis to one period (for linear prediction) or more than one period (from 4 to 5 for FFT analysis).

The Channel module also includes simple noise, impulse signal, and oscillators generators. These generators may be used for example to build a source-filter synthesizer.

II.3 Analysis Module

One of the main uses of a phase vocoder is to allow sound processing directly in the spectral domain. For this purpose, we have designed an analysis module which transforms signal windows, provided by the channel module, into spectral data.

There are three spectral data types: short-time Fourier Spectrum (STFT), spectral envelopes, and formants (frequency, bandwidth and amplitude). In this paper we define a spectral envelope as a smooth function, the value of which, in a given frequency band, is the maximum of the STFT. Frequency bands are defined according to a certain frequency resolution which separates the fine and gross structure of the STFT.

The Analysis Module controls the choice of analysis window, number of channels, and analysis type. There are typically a number of optional parameters associated with an analysis type (e.g. number of poles for linear prediction analysis).

The analysis module allows the use of five types of spectral envelope estimators. These estimators differ with respect to efficiency and the assumptions they make about a sound. The first two types are well-known [8] autocorrelation and covariance linear prediction techniques which work well on low frequency sounds. The third type is the "smoothed cepstrum" [6] which gives a smoothed spectral shape corresponding more closely to a mean of the spectrum than to a spectral envelope. The fourth is an iterative procedure called "true-envelope" [5] which starts from the smoothed cepstrum and results in a spectral envelope. This method works well but is rather slow. The fifth method is the discrete cepstrum.
which computes the spectrum envelope as a linear combination of sinusoids passing through the peaks of the spectrum. The third and fourth methods require knowledge of the fundamental frequency of the sound.

The spectral envelope may be represented by its samples or may be coded in formats for specific processing (changing a male voice into a female one for example). It may also be coded as linear predictive reflection coefficients to drive lattice filters.

II.4 Filtering Module

The filtering module processes the spectral data. This module provides seven filtering modes which differ in their manner of structuring the frequency response of the filter and in their types of control. These controls may dynamically evolve in time and are automatically interpolated between successive time samples.

The seven filtering modes are: the band filtering mode which suppresses or attenuates selected frequency bands of the spectrum defined by band pass reject limits. The gated mode which uses a frequency response defined by its samples. The break-point mode which uses a frequency response defined by a break-point function. The transfer function mode which defines the frequency response by its transfer function (poles and zeros). The second order filter bank mode which is very useful because it expresses the frequency response in terms of the intuitive format parameters. The harmonic filter mode which allows processing of individual harmonics of a sound. The surface mode which allows for attenuation of delimited areas of the time frequency domain. This mode can be driven by the graphical interface of the signal editor of the Station Musicaire IRCAM [3].

II.5 Combination Module

This module combines two sets of data arising from two different sounds and produces a new set of data. The process occurs in one pass. The module provides for a choice between three combination types such as cross-synthesis, mixing and ring-modulation and it manages the corresponding controls. The cross-synthesis mode filters a sound using the spectral envelope of another. In this case one may use the FFT analysis mode for the first sound and a spectral envelope estimator for the second. The mixing mode linearly combines two sounds, which have been separately filtered, in one pass. This is useful in applying complementary time varying filters to two sounds which are then mixed. Another example of this mode is a source filter synthesis application using separate filters for the noise and harmonic parts of a sound. The harmonic part is produced using the first voice by filtering an impulse signal, and the noise part is produced by filtering white noise in the second voice. Both signals are then added. The ring modulation mode can also be used in combination with the harmony filtering mode (see Sect. II-4). It gives a more controllable result by selecting desired partials.

This module is dramatically modified in the extended version of SVP (see Sect. III).

II.6 Synthesis Module

This module resynthesizes a sound from its spectral data. It also controls the temporal compression or dilation of the sound. At this time, there is only one synthesis type which was overlap-add and an inverse Fourier transform. But other synthesis types, such as oscillator banks, may be added.

III) EXTENDED SVP

Two major ideas have led to the extension of SVP: first, moving from a two input one output system to an N input P output multichannel system. Secondly, providing an extensible combination module with an interpreter for end permitting user defined processing algorithms. (See Figure 2).

For this, we have chosen Elk [7], a Scheme-like interpreter. It has the advantage of being portable (C source code is available in the public domain). We have provided access to the entire Job library from within Elk [3]. This multichannel and open structure has several advantages. For musical applications, one can implement any algorithm to operate on any number of sounds. For example, one can progressively separate odd and even harmonics of a sound, sending them to separate output voices. For
signal processing applications, the system forms a general program to implement block-processing algorithms on arbitrary numbers of signals. We can, in fact, implement algorithms directly within the extensible combination module, entirely by-passing the analysis and filtering modules. An important aspect of this tool is to allow simple and smooth transition from research work in signal processing to music production.

REFERENCES


Figure 1: SVP Synthesizer

Figure 2: Extended SVP Synthesizer