THE CARL PHASE VOCODER

Introduction and Reference Manual
by Trevor Wishart

This program has been made available to CDP by the Computer Audio Research Laboratory at the University of California at San Diego ('CARL').
It is distributed by CDP without charge.

CDP PVOC Functions
(FFT Analysis and Resynthesis)

PVOC **ANAL**
Convert soundfile to spectral file
PVOC **EXTRACT**
Analyse, then resynthesise sound with various options
PVOC **SYNTH**
Convert spectral file to soundfile

ANA2PVX
Convert CDP analysis file (**.ana**) to PVOC-EX file (**.pvx**)
PVOCEX2
Stereo phase vocoder based on CARL pvoc (Mark Dolson)

[PVOC] **FTURANAL ANAL**
Extract spectral features from an analysis file and output to a textfile
[PVOC] **FTURANAL SYNTH**
Use spectral features data to reassemble MONO source file

SEE ALSO:
PVPLAY
Direct playback of PVOC analysis files

PVOC MANUAL (by T.Wishart, with additional material by A. Endrich):
WINDOWS – Introduction and Analysis Windows
CHANNELS – Analysis Channels
ANAL. FILE – Fequency Analysis File
FREQ. BANDS – Points and Frequency Bands
ANAL. RATE – Analysis Rate
TERMS – Summary of Terminology
BANDWIDTH – Analysis Bandwidth

THE OPERATION OF THE PHASE VOCODER (by R.W.Dobson)
A non-mathematical introduction to the Fast Fourier Transform

PVOC ANAL – Analysis: convert soundfile to spectral file

The flags mentioned refer to Old Usage pre-Release 4. The documentation often refers to the -N flag (points) and the -W flag (overlap).

Usage

pvoc anal mode *infile outfile* [**-c***points*] [**-o***overlap*]

Modes

1 Standard analysis (was the **-A** flag)
2 Output spectral envelope values only (was the **-E** flag)
3 Output spectral magnitude values only (was the **-X** flag)

Parameters

infile – the input is a MONO soundfile

Stereo or multi-channel input must first be split into separate channels using **HOUSEKEEP CHANS 2** or **CHANNELX**. The individual mono files are then PVOC-analysed and processed separately, before being re-synthesised (PVOC SYNTH) and re-combined using **SUBMIX INTERLEAVE** or **INTERLX**.

outfile – the output is an analysis file in Mode 1

In Modes 2 and 3, the output is thought to be a binary data file. These options have been retained from the original PVOC, but in fact we are unclear as to what they produce and what program could read the data! If any CDP user has made use of these options and could enlighten us further, please do so.

-c*points* – the number of analysis points (2-32768 – powers of 2 only) Default: 1024 (was the **-N** flag)

NB: *Points* is the same as the value entered for the **-N** flag in the previous versions of PVOC. You will sometimes see references to **-N** in the CDP PVOC documentation.

More points give better frequency resolution, but worse time resolution: i.e., detail is lost in rapidly changing spectra.

-o*overlap* filter overlap factor (1 – 4) Default: 3 (was the **-W** flag)

End of PVOC ANAL

PVOC EXTRACT – Analyse, then resynthesise sound with various options

The flags mentioned (-W, -C) refer to Old Usage pre-Release 4.

Usage

pvoc extract *infile outfile* **-c***points* **-o***overlap* **-d***dochans* **-l***lochan* **-h***hichan*

Parameters

infile – the input is a soundfile
outfile – the output is a soundfile
-c*points* – the number of analysis points (2-32768 (power of 2)); Default: 1024 (was the **-N** flag)

More points give better frequency resolution, but worse time resolution: i.e., detail is lost in rapidly changing spectra

-o*overlap* filter overlap factor (1 – 4) Default: 3 (was the **-W** flag)
-d*dochans* resynthesise ODD (**1**) or EVEN (**2**) channels only (was the **-C** flag)
-l*lochan* ignore analysis channels below this when resynthesising (Default: 0)
-h*hichan* ignore analysis channels above this when resynthesising (Default: highest channel)

Lochan and *hichan* refer to channels rather than analysis points. There is 1 channel for every 2 points.
Hichan should therefore not be > analysis points/2
To default to topmost channel, set at 0.

NB: If no flags are set, the output sound will be the same as the input.

See STRETCH TIME and STRETCH SPECTRUM for stretching operations.

For the various spectral transposition options, see the REPITCH group of functions: TRANSPOSE, TRANSPOSEF, COMBINE, COMBINEB.

End of PVOC EXTRACT

PVOC SYNTH – Resynthesis: convert spectral file to soundfile

*Synthesis was the **-S** flag in the Old Usage pre-Release 4.*

Usage

pvoc synth *infile outfile*

Parameters

infile – the input is an analysis file
outfile – the output is a MONO soundfile

End of PVOC SYNTH

ANA2PVX – convert CDP analysis file to PVOC-EX file

Usage

ana2pvx *inanalfile outfile.pvx*

Parameters

inanalfile – input CDP analysis file (**.ana**)
The suffix **.ana** should be included.
outfile – output (mono) PVOC-EX file (**.pvx**)
The suffix **.pvx** should be included.

Understanding the ANA2PVX Process

ANA2PVX is an easy-to-use utility for converting an existing CDP spectral file (**.ana** format), as created by **PVOC ANAL**, to PVOC-EX format. It complements the main conversion program for PVOC-EX: **PVOCEX2**.

Related functions

PVOCEX2: Stereo phase vocoder based on CARL pvoc
PVOC ANAL: Convert soundfile to spectral file (**.ana** format)
PVOC SYNTH: Convert spectral file (**.ana** format) to soundfile

End of ANA2PVX

PVOCEX2 – Stereo phase vocoder based on CARL pvoc (Mark Dolson)

Usage

pvocex2 [**-A** |**-S**] [**-f**x] [**-o***Nosc*] [**-K** | **-H** | **-R**] [**-Px** | **-Tx**] [**-Wx**] [**-N***samps*] [**-m**] [**-v**] *infile outfile*

Example command line to convert a soundfile to PVOC-EX:
pvocex2 -A one.wav oneX.pvx [using defaults]

Example command line to convert a PVOC-EX file to a soundfile:
pvocex2 -S one.pvx one.wav

Parameters

infile – input soundfile for analysis or PVOC-EX file (**.pvx**) for re-synthesis.
outfile – output **.pvx** file (analysis) or soundfile (synthesis).
Soundfile formats supported: .wav, .aiff, .aif, .afc, .aifc
N.B. the soundfile extension must be included in the filename.

Flags:
-A – perform Analysis only: output is **.pvx** file.
or
-S – perform Synthesis only: input is **.pvx file**.
or
-I – show Format info of **.pvx** file (no outfile required).
-fx – set Frame Type for analysis file:

- x = 0 = Amplitude, Frequency (default)
- x = 1 = Amplitude, Phase (= Soundhack format)
- x = 2 = Complex (real, imaginary)

NB: The -f flag is ignored unless -A is set.
-o*Nosc* – use osc-bank resynthesis using *Nosc* oscillators (default = N/2 +1)
-K – use Kaiser window
-H – use von Hann window
-R – use Rectangular window
Default: Hamming window, or window from .pvx header
-Px – Apply Pitch scaling rate x (synthesis)
-Tx – Apply Time scaling rate x (synthesis)
NB: time/pitch scaling requires the default AMP,FREQ pvx format
-Wx – Set window overlap factor (analysis) Values: 0,1,2,3, default 1
-N*samps* – Set analysis window to *samps* samples (default 1024)
-m – Write soundfile with minimum header (no PEAK data)
-v – Suppress progress messages

Understanding the PVOCEX2 Process

CDP supports two phase vocoder formats for frequency analysis files: those converted using **PVOC ANAL** (suffix **.ana**), and the PVOC-EX format (suffix **.pvx**), devised by Richard Dobson and based on WAVE_EX. (See his **PVOC-EX page** for further details.)

PVOCEX2 is a stereo phase vocoder, converting a soundfile to PVOC-EX or from PVOC-EX to a soundfile.
The **-A** (analysis) or **-S** (re-synthesis) flag must be specified in the commandline.

A third option (**-I**) gives information about a **.pvx** file.

Some points to note in connection with **.pvx** files:

- All of the CDP spectral processes support PVOC-EX; however, you cannot process stereo **.pvx** files within CDP, only mono ones
- You must supply suitable file suffixes when converting, e.g. **myfile.wav**, not just **myfile**.
- The CDP documentation tends to assume that frequency analysis files are exclusively **.ana** files.
- The key analysis parameters for PVOCEX2 are the same as for PVOC ANAL: see **-N** (points) and **-W** (overlap) flags.
- PVOC-EX is the phase-vocoder file format used throughout Csound.

The *Sound Loom* GUI has now been converted (vn. 17.04E) to handle **.pvx**, and *Soundshaper* will also be revised.

Related functions

ANA2PVX: Convert CDP analysis file (**.ana** format) to PVOC-EX format (**.pvx**)
PVOC ANAL: Convert soundfile to spectral file (**.ana** format)
PVOC SYNTH: Convert spectral file (**.ana** format) to soundfile

End of PVOCEX2

FTURANAL – Automatic feature extraction from an analysis file
Sub-Group of 2: FTURANAL ANAL FTURANAL SYNTH

Usage

fturanal NAME (mode) *infile outfile (parameters)*
where NAME can be any one of
anal synth

Type **fturanal anal** for more info on anal option.. ETC.

FTURANAL ANAL – Extract spectral features from an analysis file and output to a textfile

Usage

fturanal anal 1 *inanalfil outfeaturefil marklist* [**-r**rand]
fturanal anal 2-3 *inanalfil outfeaturfile marklist*

Modes

Extract data ...
1 From Heads, & from Tails cut to segments, size approx equal to Heads
2 From Heads and Tails, as defined by marklist
3 From Tails only.

Parameters

inanalfile – input analysis file
outfeaturefile – textfile containing spectral data
rand – randomise timing of Tail segments (Range: 0 to 1)
outfeaturefile – the outfile is a list of ...

(a) Segment start-times (+ duration of source).
(b) (Median) Frequency of 1st formant.
(c) (Median) Frequency of 2nd formant.
(d) (Median) Frequency of 3rd formant.
(e) Spectral brightness (Range 0 to 1).

marklist – a list of timemarks in source, marking (paired) Heads and Tails

E.g.: consonant onset, and vowel continuation of source.
(It is assumed that the first mark is at a Head segment.)

Understanding the FTURANAL ANAL Process

...

End of FTURANAL ANAL

FTURANAL SYNTH – Use spectral features data to reassemble MONO source file

Usage

fturanal synth 1-10 *inwavfile outfile infeaturefile* [**-s***splicelen*]

Modes

1 Reassemble in order of increasing 1st Formant
2 Reassemble in order of decreasing 1st Formant
3 Reassemble in order of increasing 2nd Formant
4 Reassemble in order of decreasing 2nd Formant
5 Reassemble in order of increasing 3rd Formant
6 Reassemble in order of decreasing 3rd Formant
7 Reassemble in order of increasing Brightness
9 Reassemble in order of increasing Formants, summed
10 Reassemble in order of decreasing Formants, summed

Parameters

inwavfile – input mono soundfile
outfile – output soundfile
infeaturefile – a list of feature values obtained from the analysis file derived from the *inwavfile*

(a) Segment start-times (+ duration of source)
(b) (Median) Frequency of 1st formant
(c) (Median) Frequency of 2nd formant
(d) (Median) Frequency of 3rd formant
(e) Spectral brightness (Range 0 to 1)

-s*splicelen* – splice length for cutting segments, in mS (Range: 2 to 15, Default: 5)

Understanding the FTURANAL SYNTH Process

...

End of FTURANAL SYNTH

INTRODUCTION

The *Phase Vocoder* is a powerful analysis/resynthesis tool with important musical applications. The *Phase Vocoder* was initially ported from the CARL computer music package to the CDP Desktop System through the efforts of Trevor Wishart, Andrew Bentley, and Keith Henderson working with Richard Orton. Later versions were produced by T. Wishart, N. Laviers, and M. Atkins. Martin Atkins has ported it to the Atari Falcon030 and PC. The following is an outline description of its operation from a technical point of view.

Windows

Traditional acoustics based on the analysis of the sounds of pitched musical instruments teaches us that a sound may be analysed into a sum of constituent sine-waves (known as partials), each having a particular frequency and a particular magnitude (amplitude or `loudness'). This is known as the spectrum of the sound. When these frequencies are integer multiples of the smallest value (e.g. 120, 240, 360, 480 etc.) the resulting sound has one specific pitch and the sine-wave components are known as `harmonics'. The relative amplitude of the harmonics is partly responsible for differences of timbre between one instrument and another.

When the partials are not related in this simple manner, the resulting sound may have indefinite pitch (like a bass drum) or appear to contain several pitches (like a bell). At the other extreme, a sound whose spectrum changes rapidly and randomly through time produces an unpitched noise spectrum (like a clash cymbal or maracas). It was also understood that the loudness contour (amplitude envelope) of a sound significantly affects our perception of its timbre, particularly such features as the time taken to reach the maximum loudness (attack).

Traditional synthesis methods were modelled on these assumptions. However, more detailed analysis using computer technology has shown us that even the spectrum of pitched instruments varies through time. For an accurate analysis of a sound we must therefore be able to analyse the spectrum of a sound at successive instants of time which we will refer to as 'windows' (to be explained in more detail below). For a good analysis such windows are of the order of milliseconds in duration.

The *Phase Vocoder* is a program which produces such a windowed analysis of a soundfile, and will re-synthesize the sound from this analysis.

In practice one cannot simply chop up a sound into chunks for analysis. The arbitrary editing involved will produce segments of sound which drop to zero suddenly from non-zero values at the edit points (like a synthetic square-wave would do), and on analysis will contain spurious partials as a result.

To circumvent this problem, the sound-segments have a special kind of envelope imposed on them before analysis. There are a number of such window-envelopes. For most musical purposes the Hamming window can be used and this is the default window available in the CDP implementation of the Phase Vocoder (see Fig. 1).

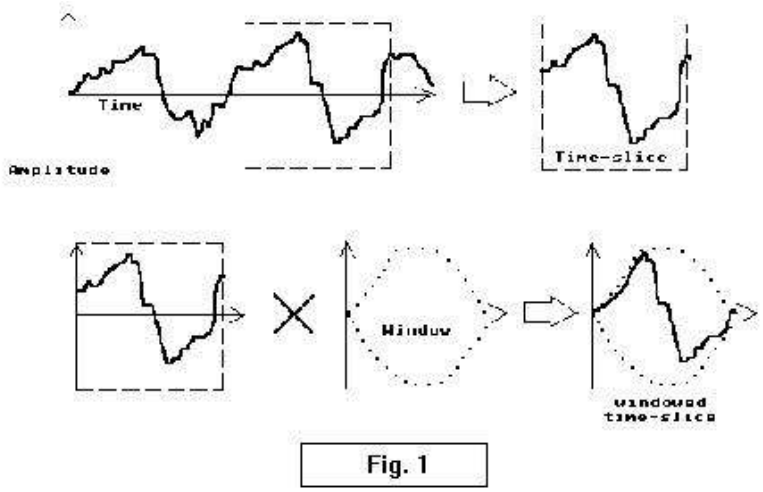


Fig. 1

PVOC, Fig. 1 – window envelopes

Channels

Within each *time*-window the *Phase Vocoder* divides the spectrum into a number of equally spaced bands known as **channels**. It then looks in each channel to see if any component of the sound is present and records its amplitude and frequency. The greater the number of channels, the more detailed the analysis.

For example, a pitched sound at 500Hz will have partials at 500 Hz, 1000 Hz, 1500 Hz (500*1, 500*2, 500*3) and so on. If there were to be 100 analysis channels, the frequency bands will be 110.25Hz wide at a sample rate of 22050 – arrived at by the calculation SR/2 divided by the number of analysis channels: (22050/2)/100 = 110.25. The various partials comprising the sound will therefore each fall into one or other of these 100 different channels of the *Phase Vocoder* analysis. Note that each channel can handle only one partial.

However a sound at 25 Hz, having partials at 25 Hz, 50 Hz, 75 Hz, 100 Hz etc will clearly have *several* of its lower partials falling within a single channel of such an analysis. In this case the *Phase Vocoder* will fail to distinguish between the several constituent partials found within a single channel and the analysis will be poor (see Fig. 2).

Therefore, the greater the number of analysis channels, the finer the resolution of the analysis. The **general rule** is that the sample rate divided by the value for **-N** should be less than the lowest pitch in the input sound. See below for more about channel width, centres and boundaries.

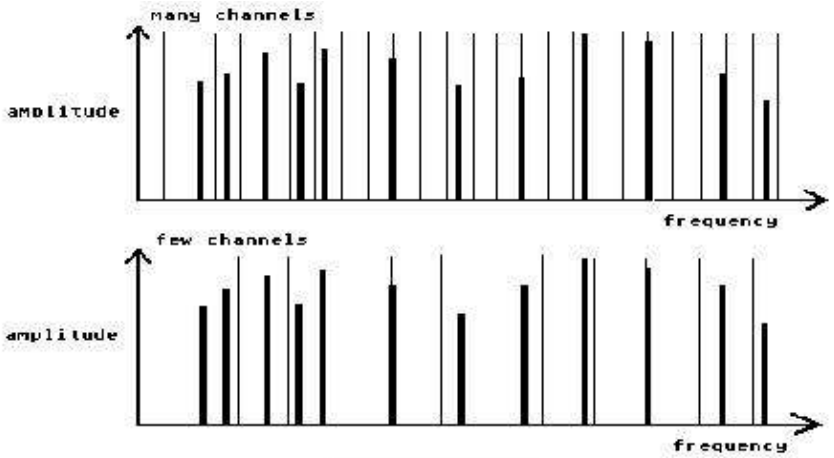


Fig. 2

PVOC, Fig. 2 – illustrating the frequency resolution of the analysis bands (channels)

Structure of the analysis file

The number of channels you wish to use may be entered with the **-N** flag of the *Phase Vocoder* program. This parameter refers to value pairs: the frequency and amplitude for each component. Thus, the number of analysis channels is only half the value you enter. Therefore, always enter twice the number of channels you want. The default value is 256. For most musical applications I would recommend entering a value of at least 1024.

Note that the Fast Fourier Transform (analysis) algorithm, which is at the core of the *Phase Vocoder* is optimised to work extremely efficiently with values which are powers of 2 (e.g. 256, 512, 1024, 2048) and you are advised to use such values unless you have a particular reason not to do so.

The following information should help to clarify some terminology and help you to use the *Phase Vocoder* more effectively, whether for musical or programming purposes.

Study example: when the soundfile sample rate is 44100 and the **-N** value is 512, what happens?

(1) The *sample rate* in the header of an analysis file is the analysis sample rate (see below) while the *analysis bandwidth* is equal to the sample rate of the original sound which was analysed divided by the value you entered with the **-N** flag (or, if you did not use the **-N** flag, the default value of 256). E.g., 44100/512 = 86.13

(2) The *Phase Vocoder* produces *two values* (amplitude and frequency) *for every analysis channel*. The convention in using the *Phase Vocoder* is to enter a value of 512, meaning 512 value-pairs per window when you want 256 equally spaced channels (amplitude and frequency value for each) and, in general, a value of 2**N** when you want **N** analysis channels. A **-N** value of 1024 is recommended.

(3) The analysis actually produces *one extra channel* centering on 0Hz at the start of each window. Thus if you enter 512 you will produce 256 channels plus 1 extra channel at 0Hz, making 257 in all.

(4) Finally, if you need to know exactly where the *channel boundaries* are, you must first calculate the *channel centres*. With a **-N** value of 512, this gives 256 channels. These 256 channels divide up the spectrum between 0Hz and half the sampling-rate (the Nyquist frequency) into equally spaced bands, and so: if the sample rate is 44100, half this is 22050, divided by 256 means that the analysis bands will centre at:-

1/2*SR*Ch_no:	0Hz	1/2 * 44100 * 1	1/2 * 44100 * 2	1/2 * 44100 * 3	..... etc
Div by Num_chnls:		256	256	256	
= channel centres:	0Hz	86.13 Hz	172.27 Hz	258.41 Hz	..... etc

The *frequency bandwidth*, i.e., the distance between these channel centres is 86.13 Hz. This is quickly calculated as SR/N – but more technically correct expressed as (1/2*SR)/(N/2). You may assume that the *channel boundaries* lie half-way between the channel centres. In the present case (see Fig. 3) this would be at:

Channel boundaries:	43.07 Hz	129.2 Hz	215.34 Hz	301.48 Hz

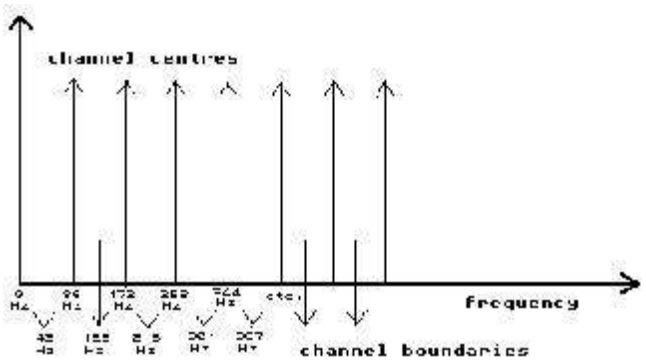


Fig. 3

PVOC, Fig. 3 – channel centres and channel boundaries

If you examine the analysis file you will discover it consists of a list of floating point numbers starting with Window 1, 0Hz channel. If you entered 512 as the **-N** value, giving you (256+1) channels, you will have 257 pairs of floating point numbers before you reach Window 2, and so on. Any number-manipulation program can be applied to this data (bearing in mind restrictions on maximum values for amplitude and frequency) so long as it generates a new analysis file of this same format. (You could generate a file with a different number of channels if you correspondingly altered the information in the analysis file header). This new file may be subsequently run through the synthesis option of the *Phase Vocoder* to generate a new soundfile.

The Phase Vocoder flags – with CDP parameter names (since Release 4) in parentheses

N (= -c*points*), F, N
sets number of channels you want in your analysis windows: (**N**/2) +1. Conversely **F** is the analysis sample rate, which is the sample rate of the input sound divided by the **N** value you want to use (frame overlap not taken into account here – see below). You cannot use both **N** and **F**.

b, e
As analysis programs always take a long time to run, you will often want to analyse only a short segment of a sound for test purposes; **b** and **e** indicate the beginning (default 0) and ending (default, end of file) samples for the analysis.

i, j, c (= **-d***dochans*)
You may use the *Phase Vocoder* as a powerful filter by excluding certain channels from the re-synthesis. Partials falling in the excluded channels will disappear; **i** indicates the lowest channel to be re-synthesized (default 0) and **j** the uppermost (default, maximum channel value) – revised as **-l***lochan* and **-h***hichan* parameters).

> Note that programs can be written to achieve more sophisticated filtering routines including cross-synthesis (imposing the channel amplitudes (which define the spectral envelope) from one sound onto a different sound); **c** is a special option resynthesizing from odd or even channels only.

A (= ANAL Mode 1), **E** (= ANAL Mode 2),**X** (= ANAL Mode 3), **S** (= SYNTH)
If you want to manipulate the sound spectrum with the program described below or with your own programs you will need to select the **-A** (= ANAL Mode 1) option to generate an analysis data file to submit to these programs. On generating a transformed analysis file, use the **-S** (= SYNTH) option to regenerate a sound file. The (**-E**) flag generates data on the spectral envelope of the sound, window by window.

T, P
The *Phase Vocoder* already contains a number of options which permit you to transform the spectrum of the input sound. For example, you may transpose the pitch of a sound without changing the duration (the **-P** option) or you may time-stretch or contract the sound without altering its pitch (the **-T** option). **T** and **P** are both multiplying factors and the default value is thus 1.0 in each case. Although these effects are more rapidly achieved with a harmonizer type of program for small degrees of stretching or compression, the *Phase Vocoder* retains much greater fidelity to the original source.
(**NB:** not implemented in Release 4 – use other functions from the REPITCH group.)

Furthermore the *Phase Vocoder* permits much greater degrees of time-stretching or pitch-transposition than any Harmonizer algorithm. For a time-stretching factor greater than ca. 8, I would recommend doing the process in 2 passes - stretch 8 times, re-synthesize the sound, analyse again, stretch and re-synthesize. Beyond a certain limit, even *Phase Vocoder* stretching will produce artifacts of one kind or another. For example, time-stretching the sound of vocal breath by a factor of 64 introduces strange glissing noise-bands (these can be heard in my piece *VOX-5*).

D, I
The decimation factor and the interpolation factor express certain mathematical relationships between sound and analysis data.

For further details, see the specialist literature. These two flags should NOT normally be used.

w

Transposing a vocal sound will distort the vowel-image rendering a text distorted or incomprehensible. This is because our perception of vocal vowels (and certain characteristics of other sounds) is related directly to the spectral envelope of the sound. If we examine the spectral envelope in any time-window we will usually find it has a number of peaks, known as formants (see Fig. 4).

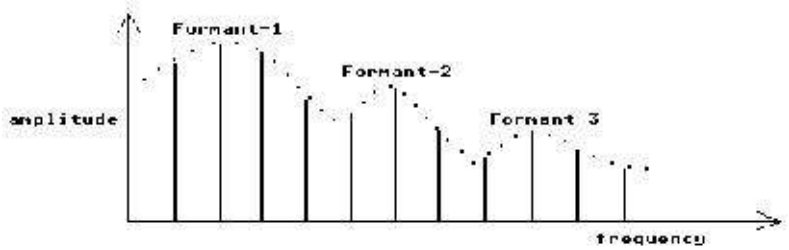


Fig. 4

PVOC, Fig. 4 – formants in the spectral envelope

Our perception of vowel 'type' is related to such formants. We are able to change the pitch of a vocal sound while retaining the vowel by using the **-w** flag. In this process the spectrum as a whole moves, but the spectral peaks remain in place centred at their original frequencies. Note that this implies that individual components of the sound change their relative amplitude to maintain the correct spectral envelope. Unfortunately when we merely transpose a sound in the *Phase Vocoder* we thereby move and stretch the spectral envelope (see Fig. 5).

In order to retain the original formants of the sound we need to 'warp' the spectral envelope back to its original position. The **-w** flag (default value 1.0) should be used to do this. Using just **-w** (i.e. with no value specified) will produce the correct warping for the pitch transposition you are using. Other specified values of **-w** may also be entered. The program SPECF below also attempts to preserve the formants of the originally analysed sound, under transformation.

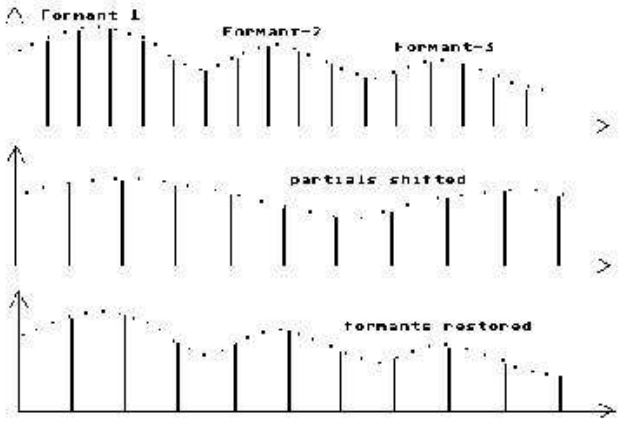


Fig. 5

PVOC, Fig. 5 – using -w to warp, retaining formants

W (= **-o***overlap*),**M,R**
For special applications we may wish to refine our analysis. Apart from increasing the number of channels we use, there are a number of other options open to us. First of all we may cause the analysis window to *overlap* by varying degrees using the **-W** flag: **W***0* is the best value for analysis and **W***3* (or **W***0*) for synthesis. You can make the analysis window *longer* (which amounts to the same thing as overlapping the windows) using the **-M** flag. The default value of **M** is (**N**-1). For good analysis results **M** should be of the order of (4 x **N**)-1. For good time-scaling it is best to let **M**=**N**-1 (but **M** = (4+**N**) will also work well). Use odd integer values for **M** : even integers will be rounded down. Flags **N** and **M** cannot be used together.

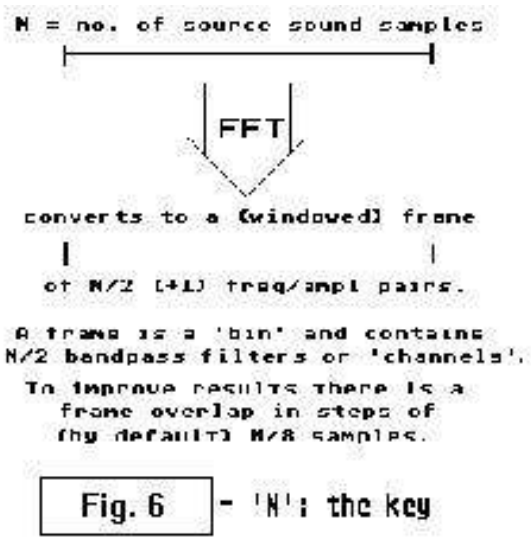
A closer look at Phase Vocoder terminology and numbers (additional material by A. Endrich)

Now let's go through the operation of the *Phase Vocoder* from various points of view in order to clarify precisely what it is doing. The purpose is to try to show more fully where various numbers come from and to connect them with the appropriate terminology. In particular, to show how the data is stored so that the meaning of some of the *Phase Vocoder* options is revealed, to show how the analysis bandwidth is calculated, where the analysis file's number of channels and sample rate come from, how to work out in which analysis channel a particular frequency can be found. The same thing is often said several times, using slightly different words, to show how the terminology is used. Thanks to the many people who helped me with this section. Also see *The Operation of the Phase Vocoder*, by Richard Dobson.

The following concise summary should become progressively clarified as this section unfolds.

The value for *points* (former -N flag) and analysis frequency bands

The **-N** flag may very well be the key to understanding what the *Phase Vocoder* does. This parameter is chosen by the user as the `number of bandpass filters' or `channels' he or she would like to have. Actually the number input for **N** is the *number of samples in the source sound to be used per analysis frame*, which results in (**N**/2) +1 channels. Fig. 6 sets out the basic outline of what happens. What it means will become clearer as we go on.



PVOC, Fig. 6 – what the value given for N does

N also sets the analysis channel bandwidth: SR/**N** = bandwidth. SR is sample rate. For example: 22050/512 = a bandwidth of 43.07Hz.

Let's go through this again a little more slowly

- There is a 1-to-1 match between data in and data out. That is, for every sample of source soundfile, there is *Phase Vocoder* analysis data.
- Two source sound samples are used to create two data of analysis (the frequency/amplitude data pair). This is why it handles up to the Nyquist frequency (ie SR/2) and not beyond it.
- The FFT analysis produces a complex number with real and imaginary parts (because it is working in three dimensions). This result is then converted to the amplitude and frequency data (pair) placed in the analysis file.
- The value for **N** input by the user is actually the number of source sound samples used for each analysis frame. In doing so, the user determines the number of analysis channels, which is (**N**/2) +1 because each pair of source sound samples (time domain) results in a frequency/amplitude pair (spectral domain). The '+1' refers to the extra channel at 0Hz.

Analysis rate: the 'sample rate' of the analysis file:

Each analysis frame uses **N** samples from the source sound. Therefore each second of analysis would use SR/**N** frames if the frames did not overlap.

Example: 22050/512 = 43.07 – that is, 512 samples per frame divided into the total number of source samples per second = 43.07 frames per second. This is the 'analysis sample rate' when there are no overlaps.

However, in order to improve the analysis, *frames are made to overlap*, normally by 1/8th the length of a frame (see Fig. 7). Thus the frame position is moved on in steps of **N**/8 samples (e.g., 512/8 = 64). The analysis `sample' rate can be described as how many of these steps of 64 samples will fit into one second of sound, or, expressed another way, how many frames will be produced when moving through the file in steps of 64 samples: e.g., 22050/64 = 344 analysis frames per second; this is the 'analysis sample rate'. More simply, 43.07 frames * 8 steps = 344.56 frames per second. Thus the 'sample rate' of the analysis file is really a 'frame rate', and each frame contains (**N**/2) +1 channels. For convenience, the decimal portion of these calculations is often omitted.

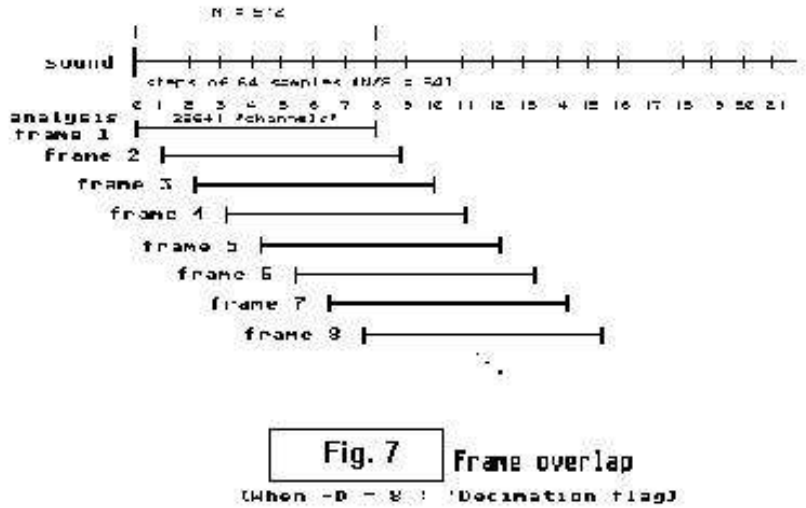


Fig. 7 Frame overlap

PVOC, Fig. 7 – analysis sample rate = frame rate per sec * number of overlaps per sec

Summary of Terminology:

points (N)

The usage refers to this parameter as the 'number of bandpass filters'. The number **N** is actually the number of samples used per analysis frame from the source sound (time domain), and the number of bandpass filters (or 'channels') is the number of data pairs that result in the frequency or spectral domain after the FFT analysis. This is why the actual number of filters or channels is (**N**/2) + 1.

frame

What, then, is a 'frame'? The frame is the basic unit 'time-slice' of analysis. It contains the data for the **N**/2 bandpass filters or channels. A `**bin**' contains the data for *one of these channels*. The frame is windowed. If **N** = 512, there will be data for 256 channels in each frame: ie 256 data pairs (frequency and amplitude) stored sequentially, plus one extra channel at 0Hz.

> Thus the amplitude value of the waveform sample (time domain) has been turned into a frequency/amplitude data pair (spectral domain). Note that each of the 256 bandpass filters or channels has only two values in it: i.e., each can only 'capture' one partial.

step

The number of samples the analysis is moved moved forward in the source sound before creating another frame. This step is by default N/8, but the 8 can be altered by the user. The parameter option to do this is the -D flag, referred to as 'decimation'. The word 'decimation' doesn't seem so odd when one realises that the larger the value for D, the smaller the step will be. The default, 8, is in fact the largest allowed.

window

refers to the fact that the bin or frame has been shaped to avoid glitches. See Fig. 1.

*RECAP: **N** is the number of source sound samples analysed per frame, yielding 256 channels +1 of analysis data in a frame or window. There is by default a frame overlap of **N**/8 samples, meaning that every sample contributes to 8 analysis frames. The amount of overlap can be changed, though this is not normally done. There are therefore SR/**step** samples per second or SR/**N***8 frames per second, which is the analysis `sample' rate.*

The analysis 'bandwidth':

SR/2 (the Nyquist frequency) divided by N/2 (actual number of bandpass filters) yields the range of frequencies handled by each filter band. Example:

> If the sample rate is 22050, half the sample rate is 11025 samples. When **N** = 512, N/2 = 256; 11025/256 = 43.07Hz per analysis channel. This can be thought of as the frequency resolution of the analysis. One doesn't divide by 257 here because 257 is the number of channel centres.

Now let's dig deeper still with few more questions.

What does the Nyquist frequency tell us?

> First of all, the point of the Nyquist frequency in this context is that it tells us what frequency range is defined by a given sample rate. The Nyquist frequency is simply SR/2: half the sample rate. Thus, the Nyquist frequency for a sample rate of 22050 samples per second is 11025 (Hz). This is the highest frequency which can be articulated at this sample rate. Put another way, a sample rate of 22050 defines a frequency range from 0 to 11025 Hz.

When the Phase Vocoder divides up the sound into a number of equally spaced analysis bands or 'channels', it is dividing up the frequency range defined by the sample rate. This is why, when we're calculating the analysis bandwidth, we work with the Nyquist frequency, SR/2.

*How do we know how large a value to enter for **points (N)** to analyse a certain sound?*

This is answered by the section on Channels above. Here's some more ideas on this subject.

The analysis bandwidth is the range of frequencies covered by each bandbass filter or analysis channel. It is calculated by dividing the Nyquist frequency by half the value you provide for **N**: SR/2 divided by **N**/2.

Some examples:

SR	**N*val***	**NyqFrq**	**N/2**	**Analysis Bandwidth**
22050	512	11025	256	43.07Hz
44100	512	22050	256	86.13Hz
48000	1024	24000	512	46.88Hz
22050	1024	11025	512	21.53Hz
22050	2048	11025	1024	10.77Hz

The partials of a 'harmonic' sound are integer multiples of the fundamental, meaning that each harmonic will be equally spaced by the number of Hz of the fundamental itself: 110 * 1 = 110, 110 * 2 = 220, 110 * 3 = 330, 110 * 4 = 440, etc.

The rule of thumb here therefore, is to make sure that the analysis bandwidth is smaller than the fundamental. Be especially careful when analyzing low sounds that you use a sufficiently large value for **N**. Why? – so that each partial of the source sound will have a separate analysis channel; otherwise, several partials will be lost.

It is important to realise that the above rule of thumb is really valid only for 'harmonic' sounds, sounds in which all or most of the significant partials are integer multiples of the fundamental. Many of the sounds we work with are in fact rather complex, from mildly inharmonic sounds to those containing quite a bit of 'noise' (fairly random combinations of partials). These complex sounds will, of course, also require a fine resolution, such as a **points (N)** value of 2048 (1012 + 1 analysis channels).

It might prove interesting however, to use a low **N** value with a low or complex sound, knowing that you are going to lose data, to create gross approximations, and possibly create some sonic artifacts: sound effects not in the original sound, but resulting from the poor analysis.

In working with the *Phase Vocoder*, it would be useful to have a handy reference chart in the form of a piano keyboard or list of MIDI notes with the appropriate frequency in Hz. It would be easy to work this out using the CDP program INFO UNITS (was MUSUNIT). You'll also find a chart like this on p.21 of *The Science of Musical Sound*, by John R. Pierce.

What about the numbers shown in the soundfile directory display?

The two numbers which may look a bit odd are the one for the number of channels and the one for the sample rate.

The figure shown as the 'number of channels' is easy: it's the value you enter for **N** plus 2 because an extra (real) channel is placed at 0Hz.

Therefore, when you enter 512, the number or channels is given as 514, etc., and the actual number of analysis channels is in this case 257. The use of **N**/2 to calculate bandwidth still seems to be valid, however.

The figure shown as the 'sample rate' for an analysis file has been explained above.

How do we relate bandwidth to channel centres and boundaries?

So far, we have looked into how the analysis frequency bandwidth is affected by the value we give for **N**, why the number of channels and the sample rate for analysis files appear as they do in the soundfile directory display. Try performing these calculations with different values for the sample rate and for **N** and check the results against what actually appears in the soundfile directory display when you actually run the *Phase Vocoder* with these values.

We can now use what we've learned from the above discussion to gain a deeper understanding of two more aspects of the analysis channels themselves: `centres' and 'boundaries'. Given a sample rate of 44100 and an **N** value of 512, the analysis bandwidth will be 86.13Hz. We're using the same numbers as on p. 3, above. This means that channel centres will occur at every 86.13Hz, starting with 0. The channel boundaries lie half way in between.

Thus the pattern is:

Channel No.	Low boundary	Centre	High boundary	Comments
0	0	0	43	(This is the extra channel, with boundaries from 0 - 43 to 0 + 43)
1	43	86	129	(86 - 43 ... 86 + 43)
2	129	172	215	(172 - 43 ... 172 + 43)
3	215	258	301	(258 - 43 ... 258 + 43)
4	301	344	387	(344 - 43 ... 344 + 43)

As shown in the section on the `structure of the analysis file' (also see Fig. 3 above).

This shows us what's going on, but of course it's a very tedious way of determining in which analysis a given frequency in the source sound might appear and this is information which we will often need.
(For example, the former programs SPECE, SPECF and SPECSH had a parameter called *fdcno* ('frequency divide channel number'), i.e., the number of the channel in which a certain frequency appears, above or below which we want something to happen – the newer versions have built this calculation into the software.)

This information can be acquired by running the utility program PITCHINFO CHANNEL. The appropriate channel number is output by the program. You can also enter a channel number and find out which frequency it will contain.

But in an effort to take another step in understanding this powerful software, let's review how a particular *fdcno* is calculated.

1. We already know how to calculate the analysis channel bandwidth: the Nyquist frequency (i.e., 1/2 SR) divided by the number of analysis channels (**N**/2).
2. We then find the channel number we need simply by dividing the frequency we want to use by the analysis channel bandwidth, and rounding up or down depending on whether the decimal part of the answer is above or below .5. This rounding handles the presence of the additional analysis channel at the beginning.

To illustrate with numbers, when SR = 22050, **N** = 512, and the frequency in question is 112Hz:

- 11025Hz divided by 256channels = 43.07Hz bandwidth per channel
- 112Hz (frequency we want to use)/43.07Hz bandwidth per channel = channel number 2.6, which rounds to 3 (only integers make sense as channel numbers).
- Thus the frequency 112Hz will be found in the 3rd analysis channel. The following chart confirms this:

Chan No.	**Low boundary**	**Centre**	**High boundary**	**Comments**
0	0.0	0.0	21.53Hz	(The extra channel)
1	21.53	43.06	64.59Hz	
2	64.59	86.12	107.65Hz	
3	107.65	129.18	150.71Hz	
4	150.71	172.24	193.77Hz	

Conclusion

We have traced what happens to the **source samples** as they are analysed, put into **bins** containing one channel each; then **N**/2 channels are assembled into a **frame**, which is **windowed**, and 8 frames **overlap** if the step by which the analysis moves on is 1/8th of a frame. The **analysis sample rate**, which appears in the sample rate part of the soundfile display, is therefore very different from the sample rate of the source sound because it is really a **frame rate**: the number of frames per second -- which must take account of the frame overlap.

The discussion of **channel centres**, **boundaries** and **bandwidth** helps us understand how a given analysis may or may not effectively capture the frequency information of a given soundfile. This information is also relevant to understanding what is meant when one of the spectral manipulation programs does something with channels, such as select the loudest component after comparing equivalent channels in several analysis files. Each channel will contain only one partial, so if there are 5 soundfiles, only one partial will 'go through to the next round': the partials belonging to 4 of the soundfiles will be dropped.

The way the frequency-amplitude information is derived from the amplitude-time information is a complicated process, with many ambiguities and approximations. Many things can be happening in the analysis bandwidth which the software may not deal with as intelligently as it might. This is one of the reasons why experimentation and careful listening are very important aspects of working with the spectral dimension.

Richard Dobson's text *The Operation of the Phase Vocoder* takes us deeper into these uncharted but vitally important areas, but the most beneficial of all will be your own exploration of these programs. These is no substitute for entering into the psycho-acoustic experience of the results as a discerning and practiced musician.

Last Updated 27 Apr 2023
Documentation: Trevor Wishart and Archer Endrich; revised R. Fraser