CDP Spectral PITCH (& Harmony/Freq) Functions

(with Command Line Usage)

Functions to Create PITCH (& Harmony/Freq) Relationships Between Partials

(Names in brackets mean that these are separate programs. The others are sub-modules of PITCH.)

ALTHARMS
Delete alternate harmonics
CHORD
Transposed versions of the sound are superimposed onto the original
CHORDF
Transposed versions of the spectrum are superimposed within the existing spectral envelope
OCTMOVE
Octave transpose without formant shift
PICK
Only retain channels which might hold the partials specified
TRANSP
Shift pitch of (part of) the spectrum
TUNE
Replace spectral frequencies by harmonics of specified pitch(es)
[TUNEVARY]
Replace spectral frequencies with the harmonics of specified pitch(es)

See also:
TRANSPOSE
Transpose spectrum (spectral envelope also moves)
TRANSPOSEF
Transpose spectrum, retaining original spectral envelope

PITCH ALTHARMS – Delete alternate harmonics

Usage

pitch altharms mode *infile pitchfile outfile* **-x**

Modes

1 delete **odd** harmonics

This usually produces a transposition an octave higher, with no formant change.

2 delete **even** harmonics

Parameters

infile – input analysis file made with PVOC
pitchfile – binary file data file

The *pitchfile* must be derived from the *infile* by running **REPITCH GETPITCH**

-x alternative spectral reconstruction

Understanding the PITCH ALTHARMS Process

To run this function, you must first extract the pitch information from the *infile* by running **REPITCH GETPITCH**. When you run PITCH ALTHARMS, it first automatically removes all the non-harmonic partials from the *infile* (the SPEC BARE process is built into it). Using the combination of this harmonic data and the pitch information in *pitchfile*, it can then remove the odd or even partials, depending on which mode you choose.

Using Mode 1 to remove the **odd** partials makes the transposition without changing the spectral contour: the formants are preserved.

Removing the **even** partials will timbrally colour the original sound, also without changing the spectral contour.

Musical Applications

A reference point for this program could be the difference between the timbre of a clarinet and that of an oboe. The sharper tone of the oboe is a result of the predominance of odd harmonics. So PITCH ALTHARMS is all about altering the tone of a sound. It does this without altering the formants, i.e., the frequency regions with the highest amplitude, so the recognisability of the sound remains, but the tone changes.

It is likely that this function will produce more interesting results the more the source sound is rich in harmonics.

End of PITCH ALTHARMS

PITCH CHORD – Transposed versions of the spectrum are superimposed onto the original

Usage

pitch chord *infile outfile transpose_file* [**-b***bot*] [**-t***top*] [**-x**]

Parameters

infile – input analysis file made with PVOC
outfile – output analysis file
transpose_file – a file of (possibly fractional) semitone transposition values
-b*bot* – bottom frequency, below which data is filtered out
-t*top* – top frequency, above which data is filtered out

bot and *top* may vary over time

-x for a fuller spectrum

Understanding the PITCH CHORD Process

The *transpose_file*, *bot* and *top* parameters are handled as in **PITCH CHORDF** below.

The difference here is that the formants are not extracted. This results in an increase in the inharmonic relationships among partials.

Musical Applications

The process enriches the sound timbrally, though also with a certain degree of resonance resulting from the presence of the 'chord'.

End of PITCH CHORD

PITCH CHORDF – Transposed versions of the spectrum are superimposed within the existing spectral envelope

Usage

pitch chordf *infile outfile* **-f***N* | **-p***N* [**-i**] *transpose_file* [**-b***bot*] [**-t***top*] [**-x**]

Parameters

infile – input analysis file made with PVOC
outfile – output analysis file
-f – extract formant envelope linear frequency-wise, using 1 point for every *N* equally-spaced frequency-channels
-p – extract formant envelope linear pitchwise, using *N* equally-spaced pitch-bands per octave
-i – quicksearch for formants (less accurate)
trans_file – a file of (possibly fractional) semitone transposition values
-b*bot* – bottom frequency, below which data is filtered out
-t*top* – top frequency, above which data is filtered out

bot and *top* may vary over time

-x for a fuller spectrum

Understanding the PITCH CHORDF Process

A brief discussion of the **extraction options** can be found in the documentation for FORMANTS GET.

The *bot* and *top* parameters allow the user to optimise the operation of PITCH CHORDF by focusing on the main part of the sound. The output of **PITCHINFO INFO** might be useful for this, as it provides data on the pitch range of the sound. (Remember that you need a binary pitch file as produced by **REPITCH GETPITCH** as an input for **PITCHINFO INFO**.)

The *transpose_file* is written as a series of semitone transposition values. 0 is used for no transposition and the 'chord' is built up from the bottom. The values can be written on separate lines or in a row, separated by spaces. Here's an example:

```
 0
 3
 4.5
 6
 7
10
12
```

OR:

```
0 3 4.5 6 7 10 12
```

If the chord began on C, for example, this chord would be C-Eb-E¼#- F#-G-Bb-C^{1}.

Musical Applications

This function provides a way to tune a sound to a chordal harmony. It is a spectral equivalent of a harmoniser: i.e., a multi-pitch shift which preserves formant information. This makes it especially suitable for the processing of vocal sounds. The net result is that the original sound becomes a chord of arbitrary dimensions.

Once a sound is tuned to a chord, it can be developed in various ways. To explore this, you might try tuning a sound with the *transpose_file* above, then stretching the resulting file x3 with STRETCH TIME, and then sustaining the data of the stretched sound with FOCUS ACCU, giving a value of 0.1 for *decay*.

End of PITCH CHORDF

PITCH OCTMOVE – Octave transpose without formant shift

Usage

pitch octmove 1–2 *infile pitchfile outfile* **-i** *transposition*
pitch oct 3 *infile pitchfile outfile* **-i** *transposition bassboost*

Modes

1 transpose up
2 transpose down
3 transpose down, with bass reinforcement

Parameters

infile and *outfile* are analysis files
pitchfile binary pitch data file (.frq) containing the pitch trace of a sound

The *pitchfile* must be derived from the *infile* by running REPITCH GETPITCH, the input for which is an analysis file.

transposition an integer transposition ratio (2 is an octave, 3 is a 12th, etc.)
bassboost bass reinforcement (Range: >= 0.0)
-i quicksearch for formants (less accurate)

Understanding the PITCH OCTMOVE Process

OCTMOVE provides a straightforward way to effect an octave shift of the spectrum. Note that it does this while preserving the formant characteristics of the original.

Step One is to run **REPITCH GETPITCH**

Step two is to run PITCH OCTMOVE, using the GETPITCH *pitchfile*.

Caution: PITCH OCTMOVE can only work if pitch is satisfactorily extracted by REPITCH GETPITCH. Any source with more than one pitch in it as a time (e.g., a chord, an orchestral passage etc.) does not have 'a pitch' as far as GETPITCH is concerned. Transposition of harmonically complex sound material is best done with REPITCH TRANSPOSEF. However, there is nothing to stop you exploring what might happen by running PITCH OCTMOVE with this kind of input.

Musical Applications

The transformation here results from the preserved formants appearing in different (octave) frequency locations. (The process works by deleting alternate harmonics.) You might want to compare this result with that of **PITCH TRANSP** which does not preserve formants.

If you wanted to emphasise harmonic material, you could try running **SPEC BARE** on the *infile* first, and using the harmonics-only result as the revised *infile* for PITCH OCTMOVE.

From an idea of Miller Puckette.

End of PITCH OCTMOVE

PITCH PICK – Only retain channels which might hold the partials specified

Usage

pitch pick 1–3 *infile outfile fundamental* **-c***clarity*
pitch pick 4–5 *infile outfile fundamental frqstep* **-c***clarity*

Modes

1 Harmonic series
2 Octaves
3 Odd partials of harmonic series only
4 Partials are successive linear steps (each of *frqstep* from the *fundamental*
5 Add linear displacement (*frqstep*) to harmonic partials over the *fundamental*

Parameters

infile – input analysis file made with PVOC
outfile – output analysis file
fundamental – actual fundamental frequency in Hz of the harmonic series, or arbitrary fundamental frequency in Hz used for the calculation
frqstep – frequency step in Hz to be added to another frequency. If you are thinking in semitones or intervals, use INFO UNITS to convert to Hz.
clarity – extent to which data in other channels is suppressed. Range: 0–1. Default 1.

clarity may vary over time

Understanding the PITCH PICK Process

This function locates the analysis channels which would contain the specified sets of partials, and adjusts the frequency data in those channels onto the specified frequencies. The data in the other channels may or may not be (partially) suppressed with the *clarity* parameter.

Musical Applications

PITCH PICK will tune a spectrum onto the partial sets specified by the Mode. This tuning can be made to emerge or disappear gradually by increasing or decreasing the *clarity* factor through time.

Various degrees of data reduction are likely to appear in the various modes. Modes 1 and 5 appear to maintain a fairly rich version of the original, while Modes 2 and 3 significantly thin the sound, the latter sounding a bit 'hollow'.

End of PITCH PICK

PITCH TRANSP – Shift pitch of (part of) the spectrum, keeping harmonic relationships

Usage

pitch transp 1–3 *infile outfile frq_split* [**-d***depth*]
pitch transp 4–5 *infile outfile frq_split transpos* [**-d***depth*]
pitch transp 6 *infile outfile frq_split transpos1 transpos2* [**-d***depth*]

Example command line:

```
pitch transp 6 inf outf 880.0 7 -4 -d0.6
```

Modes

1 Octave shift up, above *frq_split*
2 Octave shift down, below *frq_split*
3 Octave shift up and down from *frq_split*
4 Pitch shift up, above *frq_split*
5 Pitch shift down, below *frq_split*
6 Pitch shift up and down from *frq_split*

Parameters

infile – input analysis file made with PVOC
outfile – output analysis file
frq_split – frequency in Hz above or below which the shift takes place
transpos – transposition above or below *frq_split* (semitones)
transpos1 – transposition above *frq_split* (semitones)
transpos2 – transposition below *frq_split*(semitones)
-d*depth* – transposition effect on source: from 0 (no effect) to 1 (full effect)

The semitone transposition values may be fractional.

Understanding the PITCH TRANSP Process

PITCH TRANSP transposes (part of) the spectrum up or down a specified number of semitones (including fractional parts of semitones). All the partials are moved, preserving the relationships among the partials: because the semitone transposition value becomes a multiplier within the program.

Musical Applications

What this means musically, is that some of the timbral character of the original remains with the PITCH TRANSP process (relationships among partials maintained), and some of the timbral character is lost (formants not preserved).

The output of **PITCHINFO INFO** can be useful because it returns the maximum, minimum and mean pitch of a soundfile. Remember that it works on a binary pitch data file as produced by **REPITCH GETPITCH**.

It would be useful to compare, experimentally, the results of PITCH TRANSP with **PITCH OCTMOVE** and with **STRETCH SPECTRUM**. PITCH OCTMOVE preserves formants, whereas PITCH TRANSP does not. And with PITCH TRANSP, the relationships between the partials are maintained, whereas with STRETCH SPECTRUM, the stretch factor is applied to the uppermost (or lowest) channel only, and all the partials inbetween are rescaled by intermediate amounts, depending on their relative position. Try listening to these subtle differences: 1) formants preserved (OCTMOVE), 2) formants not preserved, but harmonic relationships maintained (TRANSP), and 3) rescaling the partial relationships (STRETCH SPECTRUM).

End of PITCH TRANSP

PITCH TUNE – Replace spectral frequencies by harmonics of the specified pitch(es)

Usage

pitch tune mode *infile outfile pitch_template* [**-f***focus*] [**-c***clarity*] [**-t***trace*] [**-b***bcut*]

Example command line:

```
pitch tune 2 inf outf tunemidi.txt -c0.8 -ttunetrce.txt
```

Modes

1 enter *pitch_template* data as frequency (in Hz)
2 enter *pitch_template* data as (possibly fractional) MIDI values

Parameters

infile – input analysis file made with PVOC
outfile – output analysis file
-f*focus* – determines the degree of focusing of the partial pitches onto the template (Range: 0 to 1. Default is 1)
-c*clarity* – determines the degree to which non-template partials are suppressed (Range 0 to 1. Default is 0)
-t*trace* – specifies the number of (window-by-window) most prominent channels to be replaced by the template frequencies
-b*bcut* – ignore frequencies below *bcut*, the Bass cutoff frequency

All parameters may vary over time

Understanding the PITCH TUNE Process

This program provides the most elegant way to tune a spectrum to a specified pitch set, or 'harmony'. Every component of the analysis is tuned to either one of the specified pitches, or to one of their harmonics. The final sound is a very 'natural' sounding resonance based on the specified chord. carrying the spectral articulation of the original sound. The process works best on unpitched, pitch-unspecific, or noisy materials, but can be applied to any sound.

The specified pitch set is best situated within the main pitch range of the sound: in the region of the fundamental(s). The simplest way to do this is with a tuning fork. **SPECINFO REPORT** Mode 3 will also list the frequencies of the peaks in the evolving spectrum. And if there is sufficient pitch material to get a 'reading', **PITCHINFO INFO** can give a useful overview of the maximum, minimum and mean frequencies. **NB:** the PITCHINFO functions will not work with binary pitch data files (produced by REPITCH GETPITCH) if the **-z** flag was used.)

The data is entered on the same line separated by spaces, or as a vertical list. The **Equivalent Pitch Notations** chart, arranged in octaves (e.g., C-x : 60), can be handy when entering pitch data in MIDI values. The 'x' indicates which octave, starting from the bottom of the MIDI range.

Musical Applications

This tuning process can be used simply to colour or sound, to build harmonic structure into the composition, or to make direct links with other pitch material, such as from live acoustic instruments. Note that the 'specified pitch set' can be arbitrarily constructed.

ALSO SEE: TUNEVARY, PITCH PICK

End of PITCH TUNE

TUNEVARY – Replace spectral frequencies with the harmonics of specified pitch(es)

Usage

tunevary tunevary *inanalfile outanalfile pitch_template* [**-f***focus*] [**-c***clarity*] [**-t***trace*] [**-b***bcut*]

Example command line to create spectral tuning effects:

```
tunevary tunevary in.ana out.ana ptemplate.txt ...
```

Parameters

inanalfile – input analysis file
outanalfile – output analysis file
pitch_template – textfile containing pitch data as a list of lines, each line containing a time followed by a (possibly fractional) MIDI pitch value(s).

- Times must start at zero and increase from line to line.
- Lines must have equal numbers of entries, but entries can be duplicated.
- Pitches are interpolated over time.

Example of pitch template file:

```
[time  MPVs]
0.000  60  65  70  75  80
1.000  68  75  80  87  92
2.999  68  75  80  87  92
3.000  64  70  72  78  81
```

-f*focus* – determines the degree to which partial pitches are focused onto the template. Range: 0 to 1. This parameter can be a single value or a *time focus* breakpoint file. The Default value is 1.
-c*clarity* – determines the degree to which non-template partials are suppressed. Range: 0 to 1. This parameter can be a single value or a *time clarity* breakpoint file. The Default value is 0.
-t*trace* – specifies window-by-window the number of most prominent spectral channels to be replaced by the template pitches. (Range: 1 to number of channels)
-b*bcut* – ignore any pitches below *bcut*, the Bass cutoff frequency given in Hz.

All parameters may vary over time.

Understanding the TUNEVARY Process

This complementary spectral program builds on and combines aspects of **PITCH TUNE** and **FILTER VARIBANK**. Thus it can tune a spectrum to the specified pitches (given that those pitches are in the analysis file, otherwise, it skips or approximates them), AND it can do so in a time-varying way. Thus the pitch template file also contains times. TUNEVARY can therefore be seen as an extension of **PITCH TUNE**.

Musical Applications

Same as **PITCH TUNE**. The output can also be mixed with that of **FILTER VARIBANK**, especially where the latter has used similar or identical pitch-data.

End of TUNEVARY