Composers' Desktop Project

CDP SPECNU: Cleaning and other Functions

(with Command Line Usage)

Functions to clean up analysis files

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

The SPECNU Group includes command line programs to perform various audio cleaning procedures. The *Sound Loom* GUI contains a comprehensive **Cleaning Kit** that employs these programs as well as others, conveniently assembling a whole range of functions for audio cleaning in one place. This is described in the **Technical** section below.
Release 7 has added two other functions to the group: **RAND** and **SQUEEZE**.

CLEAN
Eliminate from the source file any persisting signal that falls below a threshold (defined by the noise file)
RAND
Randomise the order of spectral windows
REMOVE
Remove a pitched component from the spectrum of a sound
SLICE
Divide an analysis file into individual frequency bands, saving each as a separate analysis file
OR: Invert spectral frquencies around a given frequency
[SPECGRIDS]
Partition a spectrum into parts, over a grid
SQUEEZE
Squeeze the spectrum into a frequency range, around a specified frequency
SUBTRACT
Eliminate from the source file any persisting signal that falls below a threshold (defined by the *noisfile*) AND subtract the amplitude of the noise in the *noisfile* from any source file signal that is passed

Technical Discussion
Additional information derived from the **Cleaning Kit** in *Sound Loom*.

SPECNU CLEAN – Eliminate from the source file any persisting signal that falls below a threshold (defined by the noise file)

Usage

specnu clean *sigfile noisfile outanalfile persist noisgain*

Parameters

sigfile – analysis file of the source sound to be cleaned
noisfile – analysis file cut from the source sound, containing a sample of the noise (only) to be eliminated. Try to find a section containing only that noise, e.g., at the beginning or end of the sound.
outanalfile – resulting cleaned sound
persist – minimum time (in ms.) for which the clean signal must persist *above the threshold level* if it is to be retained (Range 0 to 1000 ms.)
noisgain – multiplies the threshold noise-levels found in the *noisfile* by an amount you specify, before they are used for comparison with the source file. (Range: 1 to 40)

Both input files are analysis files. *Noisfile* is a sample of noise (only) from the *sigfile*.

Understanding the SPECNU CLEAN Process

This process enables sources (once analysed) to be cleaned, and is similar to the earlier version of SPEC CLEAN. It differs from SPEC CLEAN by having a *persist* parameter.

Thus with SPECNU CLEAN you can specify how long a signal must *persist* before you will accept it as valid data (rather than simply noise to be eliminated).

This avoids the 'bubbling' problem of the original SPEC CLEAN. With SPEC CLEAN, when a signal was near the noise threshold, the spectrum might be judged as acceptable signal, or as noise, from one window to another, possibly changing rapidly between (accepted) signal and (eliminated) noise. The result of this switching on and off of the signal in the output, is a 'bubbling' effect in the sound. The *persist* parameter ensures that the signal does not bubble in and out in this way.

Musical Applications

This process is intended for cleaning up recordings made in noisy environments. It may be used alongside various editing and filtering processes, to clean a source sound bit by bit.

End of SPECNU CLEAN

SPECNU RAND – Randomise the order of spectral windows

Usage

specnu rand *inanalfile outanalfile* [**-t***timescale*] [**-g***grouping*]

Example command line to randomise the order of spectral windows:

```
specnu rand inanlfile outanalfile 0.5 50
```

Parameters

inanalfile – input analysis file
outanalfile – output analysis file
-t*timescale* – determines the number of windows locally-randomised. Default: all the windows are randomised
-g*grouping* – consecutive windows are grouped into sets of size *grouping*. These sets are randomised. The default grouping is one window per set.

Timescale may vary over time.

Understanding the SPECNU RAND Process

The degree of randomisation is controlled by the user. By default a set contains only one window. If all the windows are randomised, this means that the maximum randomisation is achieved. The amount of randomisation can be reduced in two ways using the optional parameters.

- Setting a *timescale* focuses the randomisation into specific time areas. Thus, as you move forward through the input file, time-specific areas will be jumbled.
- Giving a value for *grouping* > 1 keeps this number of consecutive windows the same, but then jumbles the sets of windows.

Musical Applications

SPECNU RAND provides a way of texturing a sound. Furthermore, as it is operating in the Spectral Domain, the process will create timbral variations.

End of SPECNU RAND

SPECNU REMOVE – Remove a pitched component from the spectrum of a sound

Usage

specnu remove 1-2 *inanalfile outanalfile midimin midimax rangetop atten*

Modes

1 Removes pitch and its harmonics up to a specified frequency limit
2 Removes everything besides the specified pitched material

Parameters

inanalfile – input analysis file with pitched components
outanalfile – resulting analysis file with pitched component removed
midimin – minimum pitch to remove (MIDI pitch values 1-127)
midimax – maximum pitch to remove (MIDI pitch values 1-127) (all pitches between *midimin* and *midimax* are removed)
rangetop – frequency (hz) at which the search for associated harmonics (TO BE ELIMINATED) stops
atten – degree of attenuation of the suppressed components Range: 0 to 1 (1 = maximum attenuation, 0 = no attenuation)

The MIDI range between *minimin* and *midimax* should be small.
If the range is an octave or more, the whole spectrum between *midimin* and *rangetop* will be removed.

Understanding the SPECNU REMOVE Process

SPECNU REMOVE will only work with stable, clearly-pitched components in the spectrum. It allows unwanted signals with steady pitch to be removed from a source. Deciding how many harmonics (and how large a pitch-range) should be eliminated is a matter of judgement based on listening to the results. Removing more is not necessarily better as the resulting hole in the spectrum can appear to emphasize the very pitch(band) you wanted to eliminate.

Musical Applications

This process is intended for cleaning up recordings made in noisy environments. It is particularly useful for removing pitched components of speech from unwanted speaking voices behind a recording. (Filtering processes can be used to remove unwanted sibillants). It may be used alongside various editing and filtering processes, to clean a source sound bit by bit. These various functions are integrated in the ***Cleaning Kit*** option on the *Music Testbed* of the *Sound Loom*.

Applied to a vocal sound, Mode **1** leaves a modest 'hole' in the sound. Trevor Wishart writes: "There are lots of complications in removing a pitch from a sound, which the user will discover. If you remove too many harmonics, the hole itself seems to produce the very pitch you were trying to remove! I have no idea why this is. The high frequency limit should therefore be kept *as low as possible* if you are seriously attempting to remove an existing pitch element from a sound."

Mode **2**, in removing everything else, can leave a rather pinched voice, which may have compositional uses.

End of SPECNU REMOVE

SPECNU SLICE – Divide an analysis file into individual frequency bands, saving each as a separate analysis file;
invert spectrum around given frequency (mode 5)

Usage

specnu slice 1-3 *inanalfile outanalfile bandcnt chanwidth*
specnu slice 4 *inanalfile outanalfile pitchdata*
specnu slice 5 *inanalfile outanalfile freq*

Modes

Mode 1 Moirée slice – alternate *chanwidth*-wide (groups of) channels are sent to the different output files.

For example: with *bandcnt* = 3 and *chanwidth* = 1, groups of just 1 channel width are selected cyclically, to make 3 output files:

- File 1 gets channels 0 : 3 : 6 : 9 : 12 : 15 : 18 ... etc.
- File 2 gets channels 1 : 4 : 7 : 10 : 13 : 16 : 19 ... etc.
- File 3 gets channels 2 : 5 : 8 : 11 : 14 : 17 : 20 ... etc.

With *bandcnt* = 2 and *chanwidth* = 2, groups of 2-channel width are selected cyclically, to make 2 output files:

- File 1 gets channels 0+1 : 4+5 : 8+9 : 12+13 : 16+17 ... etc.
- File 2 gets channels 2+3 : 6+7 : 10+11 : 14+15 : 18+19 ... etc.

With *bandcnt* = 5 and *chanwidth* = 4, groups of 4-channel width are selected cyclically, to make 5 output files:

- File 1 gets channels 0+1+2+3 : 20+21+22+23 ... etc.
- File 2 gets channels 4+5+6+7 : 24+25+26+27 ... etc.
- File 3 gets channels 8+9+10+11 : 28+29+30+31 ... etc.
- File 4 gets channels 12+13+14+15 : 32+33+34+35 ... etc.
- File 5 gets channels 16+17+18+19 : 36+37+38+39 ... etc.

Mode 2 Frequency band slice – equivalent to Mode **1**. The *chanwidth* is defined as frequency (in Hz) rather than as a count of analysis channels (Range: 43.066406 to 11025.0).

Mode 3 Pitch band slice – similar to Mode **2**, but the *chanwidth* is now defined in semitones, and the slices are of equal **pitch width** (rather than of equal **frequency** width, as in Modes **1** and **2**.

Mode 4 Harmonics slice – similar to Mode **2**, but each band is centered on a harmonic of the sound. The position of the harmonics is determined from the *pitchdata* textfile (possibly extracted using REPITCH GETPITCH – using Mode **2** to produce a breakpoint text file).

Mode 5 Invert spectrum – spectral frequencies are inverted, around a given frequency value.

Parameters

inanalfile – input analysis file
outanalfile – multiple output analysis files: the first file is as named, the second and subsequent files replace the last character of the first file's name with '1', '2', etc.
bandcnt – the number of output files (and the channel separation of the bands in the output sounds)
chanwidth – values differ in the various modes:

- Mode **1:** *chanwidth* is the **number of channels** in each band in each output sound
- Mode **2:** *chanwidth* is the **width in Hz** of each band in each output sound
- Mode **3:** *chanwidth* is the **width in semitones** of each band in each output sound

pitchdata – breakpoint text file containing *time pitch* value pairs
freq (Mode 5) – the frequency around which to invert all spectral frequencies.

Understanding the SPECNU SLICE Process

This process slices a signal into a number of spectral components, that can then be reassembled into the original sound, or used in some other way. The output sounds will contain a number of bands (*bandcnt*) and channel widths (*chanwidth*), depending on the user's selections, so each sound will be an amalgam of different parts of the original source.

It is not possible to predict which sound will appear higher or lower, as each sound is made up of slices taken from the whole range of the source. Thus it all depends on what the source contains. In the various band slices, there could be:

- nothing at all (don't be surprised if you get a silent outfile)
- something, but a very low level (i.e., amplitude)
- something very prominent
- something that combines the contents of another band (in the stack in question) to produce a prominent output.

In Mode **4**, the lower the pitch of the source, the more outputs are produced because there are more potential harmonics before you reach the maximum frequency limit. In fact, there can be a very large number of output files, many of which will be silent. This process does not know if there is any energy at any particular harmonic. It just tracks where the harmonics of the sound would be, and reports what it finds. A pure sine tone for example is going to produce many silent outputs. The number of silent outputs will be reduced if the sound has a very rich or bright spectrum, the program is of more use with this kind of input file.

Mode **5** does not produce multiple outputs and is, in effect, a separate function. The frequencies that make up the spectrum of a sound are inverted (turned upside-down) above and below the given frequency value, so that those that went upwards from this frequency now go down, and vice versa. A pitched sound will normally be turned into one with inharmonic partials. This is a powerful technique to create entirely new timbres from familiar sounds, and may be compared with STRANGE SHIFT, which also creates inharmonic sounds.

Musical Applications

At the moment it appears to be a way to get a number of tonally different versions of the input analysis file rather quickly. The potential for composition lies in possible recombinations of these rather unpredictable slices, especially if they are processed separately before being recombined.

To give some idea of what might be done, consider that interesting IRCAM experiment where different rates of vibrato are applied to the odd and to the even harmonics of a vocal sound. You start with the odd harmonics in the left channel and the even harmonics in the right channel. With no vibrato, and you hear a (mono) voice at stage centre. But when you apply a differentiated vibrato you hear a 'clarinet' in one channel and a 'cello' in the other. This suggests the idea of generalising this approach, applying different processes to the different harmonics, and recombining the results, to see what happens. Do you get an elaborate variation of the original sound or does the single original image split into several different images, and if so might this be musically useful?

The potentially large number of outputs suggests that it would be practical to approach using this program by means of batch processing, perhaps controlled by scripting with a batch file or higher level scripting language. Scripting multi-sound playback, for example, would enable all the outputs to be heard in sequence, and this could then be extended to eliminating silent files or processing e.g., alternate files in different ways.

End of SPECNU SLICE

SPECGRIDS – Partition a spectrum into parts, over a grid

This function is identical to SPECNU SLICE, Mode 1, as Trevor Wishart has confirmed.

Usage

specgrids specgrids *inanalfile1 outanalfilesname outfilecnt changrouping*

Example command line to partition the spectrum:

```
specgrids specgrids mysnd.ana split.ana 8 8
```
(produces multiple output files: split0.ana, split1.ana, etc.)

Parameters

inanalfile1 – input (mono) analysis file
outanalfilename – root name for a series of output analysis files
outfilecnt – the number of output spectral files to create
chan-grouping – the number of adjacent channels per group in the output spectra

NB: The product of *outfilecnt* and *chan-grouping* must be an (even) divisor of the spectral channel-count, e.g., 512.

Understanding the SPECGRIDS Process

This program enables you to separate out different parts of a sounds spectrum and place these different parts over a multi-channel rig. First it divides the spectrum of a set of disjunct spectra. This means that the input analysis file needs to have distinctly different spectral features – heard as timbral colours.

Each output spectrum will contain some selection of the spectral channels of the source, while setting the other channels to zero amplitude. Between them, the complete set of output spectra contains all the channel info from the source.

Now you are in a position to resynthesise the different segment-streams separately and distribute the resulting soundfiles over a multichannel output. Thus the spectrum of the original sound is distributed around the multichannel space.

Musical Applications

Let's look at this more closely. If the input file has 16 channels "**abcdefghijklmnop**":

- With 2 outfiles and *chan-grouping* = 1,
  the outfiles contain "**a-c-e-g-i-k-m-o**" and "**b-d-f-h-j-l-n-p**" respectively.
- With 2 outfiles and *chan-grouping* = 2,
  the outfiles contain "**ab-ef-ij-mn**" and "**cd-gh-kl-op**".
- With 2 outfiles and *chan-grouping* = 4,
  the outfiles contain "**abcd-ijkl**" and "**efgh-mnop**".
- With 2 outfiles and *chan-grouping* = 8,
  the outfiles contain "**abcdefgh**" and "**mnopijkl**".

- With 4 outfiles and *chan-grouping* = 1,
the outfiles contain "**a-e-i-m**" "**b-f-j-n**" "**c-g-k-o**" and "**d-h-l-p**".
- With 4 outfiles and *chan-grouping* = 2,
the outfiles contain "**ab-ij**" "**cd-kl**" "**ef-mn**" and "**gh-op**".
- With 4 outfiles and *chan-grouping* = 4,
the outfiles contain "**abcd**" "**efgh**" "**ijkl**" and "**mnop**".

End of SPECGRIDS

SPECNU SQUEEZE – Squeeze spectrum in frequency range, around a specified centre frequency

Usage

specnu squeeze *inanalfile outanalfile centrefrq squeeze*

Example command line to create the squeeze effect:

```
specnu squeeze inanalfile outanalfile 341 0.4
```

Parameters

inanalfile – input analysis file
outanalfile – output analysis file
centrefrq –frequency in Hz around which frequency data is squeezed
squeeze – the amount the spectrum is squeezed (< 1)

All parameters may vary over time:
breakpoint files containing *time centrefrq* and *time squeeze* data columns separated by a space or a tab.

Understanding the SPECNU SQUEEZE Process

Note that the frequency at which the squeezing is to take place can be specified (*centrefrq*). The result will be some form of intensification (aural density) in that part of the sound. The degree of density will depend on the *squeeze* factor, higher factors yielding complex 'tones' in the midst of the overall sound.

At the time of writing, the program is producing output centred around 341Hz, only. However, combined with suitable transposition(s), e.g. using REPITCH TRANSPOSE, it might still be a useful way to colour a sound.

End of SPECNU SQUEEZE

SPECNU SUBTRACT – Eliminate from the source file any persisting signal that falls below a threshold (defined by the *noisfile*) AND subtract the amplitude of the noise in the *noisfile* from any source file signal that is passed

Usage

specnu subtract *sigfile noisfile outanalfile persist noisgain*

Parameters

sigfile – input analysis file with noise to be removed
noisfile – a sample of noise, cut from the original signal, containing only (a sample of) the noise you wish to remove from the *sigfile*
outanalfile – output analysis file with noise removed
persist – minimum time (in ms.) for which the clean signal must persist *above the threshold level* if it is to be retained. (Range 0 to 1000 ms.)
noisgain – multiplies the threshold noise-levels found in the *noisfile* by an amount you specify, before they are used for comparison with the source file. (Range: 1 to 40).

NB: Both input files are analysis files.

Understanding the SPECNU SUBTRACT Process

This process is similar to SPECNU CLEAN. SPECNU CLEAN assumes that, once your desired signal has reached an appropriate level it will mask any persisting background noise. With very noisy environments, this may not be sufficient, and the entire noise signal will need to be subtracted from the source. This process is only useful where the noise in the signal is persistent (e.g. the noise of an air-conditioning unit.)

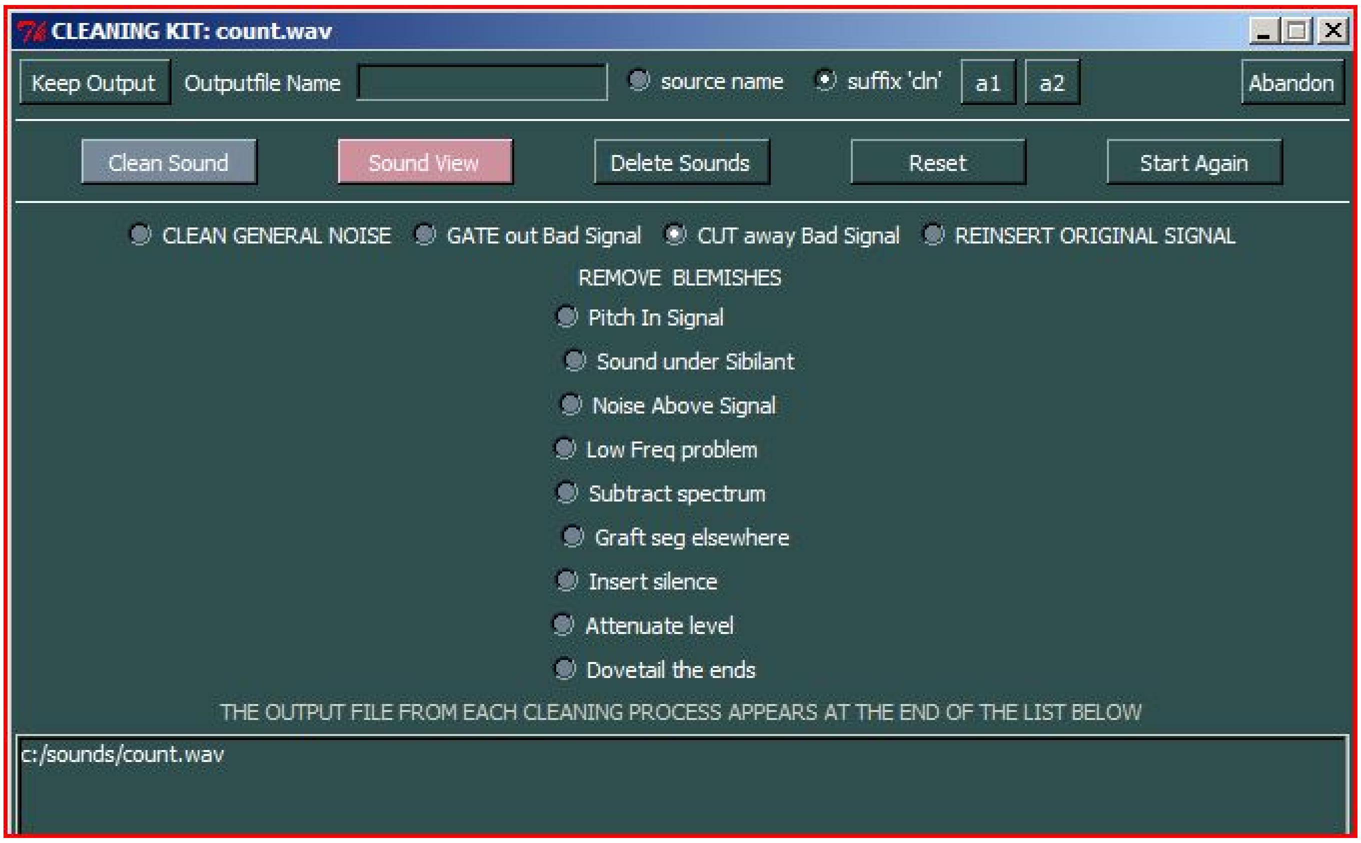
Musical Applications

This process is intended for cleaning up recordings made in noisy environments. It may be used alongside various editing and filtering processes, to clean a source sound bit by bit. These various functions are integrated in the ***Cleaning Kit*** option on the *Music Testbed* of the *Sound Loom*.

End of SPECNU SUBTRACT

Additional information about cleaning soundfiles

These four SPECNU functions and several others are integrated in the **Cleaning Kit** option on the **Music Testbed** of the *Sound Loom*. The following screen indicates what is available:



[✓ suffix 'dn']

[✓ CUT away Bad Signal]

THE OUTPUT FILE FROM EACH CLEANING PROCESS APPEARS AT THE END OF THE LIST BELOW

c:/sounds/count.wav

The way these functions relate to the SPECNU programs is hidden to the user, as the *Sound Loom* Cleaning Kit builds the SPECNU functionality into a much broader context. The input to the Cleaning Kit is a normal soundfile, not an analysis file, and when a selection is made and the CLEAN button is activated, new dialogues appear which enable you to enter data and complete the processing.

More information is included in a page from Trevor Wishart's website. We link to a current copy of it here for your convenience. This document contains a great deal of useful information about potential problem areas in sounds and how one might deal with them.

Last Updated 29 June 2015
Documentation: Archer Endrich, revised R. Fraser