CDP Functions to HILITE Spectral Data

(with Command Line Usage)

Functions which HIGHLIGHT Spectral Amplitude

ARPEG
Arpeggiate the spectrum
BAND
Split spectrum into bands and process these individually
BLTR
Time-average and TRACE the spectrum
FILTER
Filter the spectrum
[GLISTEN]
Randomly partition the spectrum into bins and play back in order
GREQ
Graphic EQ type filter on the spectrum
PLUCK
Emphasise spectral changes (use e.g., with HILITE ARPEG)
TRACE
Highlight *n* loudest partials, at each moment (window) in time
VOWELS
Impose vowels on a sound
Technical Discussion
Partials and frequencies

HILITE ARPEG – Arpeggiate the spectrum

Usage

hilite arpeg 1-4 *infile outfile wave rate* [**-p***U*] [**-l***X*] [**-h***Y*] [**-b***Z*] [**-a***A*] [**-N***k*] [**-s***S*] [**-T**] [**-K**]
hilite arpeg 5-8 *infile outfile wave rate* [**-p***U*] [**-l***X*] [**-h***Y*] [**-b***Z*] [**-a***A*]

Modes

1 ON.....Play components inside arpeggiated band ONLY
2 BOOST.....Amplify snds in band. Others play unamplified
3 BELOW_BOOST.....INITIALLY play components in & below band ONLY

THEN amplify sounds in band. Others play unamplified
(**NOT with downramp** – illogical)

4 ABOVE_BOOST.....INITIALLY play components in & above band ONLY

THEN amplify sounds in band. Others play unamplified
(**NOT with upramp** – illogical: with sin/saw startphase > 0.5)

5 BELOW.....Play components in & below arpeggiated band ONLY
6 ABOVE.....Play components in & above arpeggiated band ONLY
7 ONCE_BELOW.....INITIALLY Play components in and below band ONLY

THEN play whole sound as normal. (**NOT with downramp – illogical**)

8 ONCE_ABOVE.....INITIALLY Play components in and above arpeggiated band ONLY

THEN play whole sound as normal
(**NOT with upramp** – illogical: with sin/saw startphase > 0.5)

Parameters

infile – input analysis file made with PVOC
outfile – output analysis file
wave – **1** = downramp : **2** = sin : **3** = saw : **4** = upramp
rate – number of sweeps per second (can be < 1)
-p*U* – start_phase: range 0-1 (limited range for some cases); may not affect the sound very much
-l*X* – lowest frequency arpeg sweeps down to; Default = 0
-h*Y* – highest frequency arpeg sweeps up to; Default nyquist
-b*Z* – bandwidth of sweep band (in Hz); Default = nyquist/channel_cnt (i.e., sample rate/2/channel count)
-a*A* – amplification of arpegtones; Default = 10.0
-N*k* – nonlinear decay arpegtones; > 1 faster, < 1 slower; must be > 0 (range: 0.02 to 50 – take care with values higher than about 5, as can reduce output to silence)
-s*S* – number of windows over which arpegtones sustained: Default = 3 (high values, esp. with a long decay time, can cause amplitude overflow)
-T – In sustains, TRACK changing frquency of source (Default = retain start frquency)
-K – Let sustains run to zero before new arpegtone attack is accepted (Default: re-attack once sustains fall below current input level)

All parameters may vary over time, except for *wavetype* and *startphase*

Understanding the HILITE ARPEG Process

HILITE ARPEG means 'spectrum arpeggio' and causes a waveform at a specified frequency to sweep through the spectrum of a sound, selecting or emphasizing the partials it passes through (or which lie below it or above it), thus creating an 'arpeggiation' of the spectrum within the sound.

Note the interaction of the **-N** and **-s** parameters. The one is controlling the envelope of the decay, and the other is sustaining across windows. When a slow decay envelope (e.g., 0.2) is combined with a large number of windows across which to sustain data (e.g., > 100), amplitude overflow can occur, sometimes massive amplitude overflow if the source is loud. If PVOC comments on amplitude, take care to to read what it says before playing the sound. Values for **-a** also affect amplitude output, so when near the limit with other factors, a lower value for **-a** can be helpful.

High values for **-N** can muffle the sound, and very high values can reduce it to silence. You are recommended not to exceed a value of 7.

Rule of Thumb: increase the decay rate (higher value for **-N**) as you increase the sustain windows (**-s**).

The **-T** parameter causes arpeggios to follow (the moving) pitch content of the sound. The **-K** parameter places the attacks in relief by making sure that the sustains run to zero before a new attack begins.

The appropriate *rate* parameter value can be imagined by likening the effect to a vibrato, i.e., a low frequency oscillator. Something about 10 to 20 cycles per second is what is expected, but slower than this can be used to create glissandos within the spectrum. A single sweep through the sound can be useful, such as to create a sense of upwards transposition. CDP user Peter Karkut came up with this idea as a way of preparing a sound for a morph.

The *rate* parameter, we are told, must be less than the analysis rate. Note that this is NOT the value given for **-N** (now **-c** *points* when analyzing sound, but rather the analysis frame rate. As explained in the *Phase Vocoder* Manual, this is (SR/2)/(N/2), where SR is the sample rate and N is the value given for **-N**. The result of this calculation is then multiplied by 8, the number of window overlaps automatically made by the Phase Vocoder in order to improve the quality of the output. A typical frame rate is 344: (44100/2)/(1024/2) = 43.06;43.06 * 8 = 344. So in this case, *rate* must be < 344. In practice, a value of 100 is about the limit, but the software does not prevent exploring what happens beyond this point.

Musical Applications

HILITE ARPEG can be used to create an unusual attack for a sound object. Because the wave sweeps through all the analysis channels, it serves to bring out internal features inside the sound. Sometimes a pitch focus or harmonic colouration within the sound will result, particularly if the frequencies are being tracked (**-T**).

This process seems to work most clearly with sounds possessing strong and slowly moving partials, such as bell sounds and sounds perceived mostly as pitch (harmonic partials predominate over inharmonic partials).

Perception of the effect is also affected by the speed of the movement; it is more noticeable at slower speeds. This can be timed to give a single sweep over the duration of the sound because the *rate* can be a floating point value (> 0 but < the analysis rate). It might be interesting to try it on the result of a **SPEC BARE** process, because then the sweeping arpeggio will be emphasising the natural harmonic overtone series.

High sharp sweeps can be achieved with Mode 6, e.g., by a high value for **-h** and none for **-l**: makes the sweeps occur above the **-h** frequency. Try pushing this higher still, and playing with rates < 1.

There are many factors to play off against one another in HILITE ARPEG, so creating a series of batch files with which to explore the function would be time well spent. There are many serendipitous sounds hidden among the more obvious possibilities. It is more of the most powerful functions in the spectral set. (Ed.)

End of HILITE ARPEG

HILITE BAND – Split spectrum into bands and process these individually

Usage

hilite band *infile outfile datafile*

Parameters

infile – input analysis file made with PVOC
outfile – output analysis file
datafile contains the instructions on how to split the bands, written as (a number of) lines in the following format:

***lofrq hifrq bitflag* [*amp1 amp2* [+]*transpose*]**

lofrq and hifrq are entered in Hz and define the bottom and top frequencies of a band within the spectrum of the sound

bitflag has 4 bits (e.g., '0101' or '1000'):

bit 1 set: amplitude **change** to band: put amplitude multiplier *amp1* in the line
bit 2 set: amplitude **ramp**: put 2nd amplitude multiplier *amp2* in the line. The spectral band will change in amplitude from *amp1* to *amp2*. This is a spectral filter slope (envelope) between the stated frequency boundaries, applied to the whole length of the sound. Note that bit 1 also has to be set: e.g., 1100 for this function to work.
bit 3 set: partials **transpose**: the function expects the frequency multiplier *transpose* to be in the line. OR, if the value is preceded by a '+', the value is ADDED to the frequency in the band.
bit 4 set: transposed **partials are ADDED** to the original spectrum. Default: Replace.

amp1, *amp2* &/or *transpose* MUST be present when they are required by the *bitflag* options

amp1 and *amp2* are amplitude multipliers, used as described above

transpose is a frequency multiplier

Understanding the HILITE BAND Process

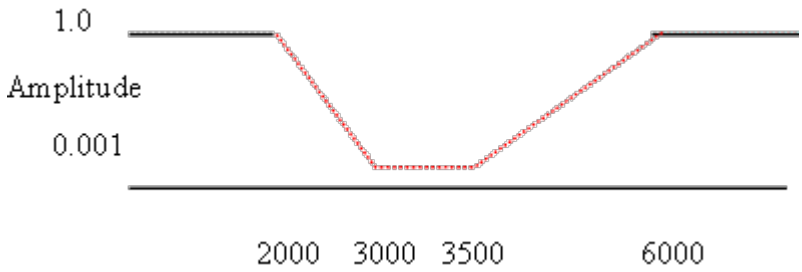
HILITE BAND used to be called SPECSPLIT, which means 'spectrum split': divide the spectrum into a number of (user-defined) frequency bands. It then carries out amplitude and/or pitch changes to the partials found in the bands. Each band may be treated differently. Typical use of HILITE BAND involves defining several data lines in order to create an arbitrary spectral filter envelope. Each data line is a single line contributing to a sort of frequency response breakpoint shape.

Musical Applications

When you set a frequency band to zero, it is filtered out (bit 1 set to 0). By retaining/emphasizing some frequency bands and zeroing others, you can focus on and enhance the bands you want to keep. This is also a way to tune or harmonize sound material. The data file for HILITE BAND can contain several lines, so several bands can be, and usually are, defined at one time.

Filter slope can be applied within frequency bands (bit 2 set and values given for *amp1* and *amp2*). In the line `3500 6000 1100 0.001 1.0`, the slope moves from 3500 Hz **upwards** towards 6000 Hz, and in the line `2000 3000 1100 1.0 0.001`, the slope moves **downwards** from 2000 Hz to 3000 Hz. The two lines as a pair give this shape:

```
2000 3000 1100 1.0   0.001
3500 6000 1100 0.001 1.0
```



Note the potential of the transposition functions (bit 3 and bit 4). Transposition by multiplication means that the harmonic relationships are maintained (logarithmic). Transposition by addition creates inharmonic relationships (linear). The superimposition of frequency bands made possible by bit 4 will have the effect of thickening the resultant sound in some way – which probably won't be very predictable.

End of HILITE BAND

HILITE BLTR – Time-average and TRACE the spectrum

Usage

hilite bltr *infile outfile blurring tracing*

Parameters

infile – input analysis file made with PVOC
outfile – output analysis file
blurring – the number of windows over which to average the spectrum
tracing – the number of (loudest) channels to retain, in each window

blurring AND *tracing* may vary over time

Understanding the HILITE BLTR Process

In each window, all but the loudest (tracing) channels are discarded (see **HILITE TRACE**. Then a proportion of the windows are discarded and replaced by values interpolated between the remaining windows (see **BLUR BLUR**. The degree of the change will depend on the amplitude differential between the loudest and not so loud channels: the louder the latter are, the more their absence will be noticed.

Musical Applications

The dual functionality of HILITE BLTR enables the sound designer to reduce the data by removing all but the *tracing* loudest channels, and simultaneously to smooth/blur the remaining data by averaging the frequency and amplitude values contained in each set of *blurring* windows. This is a good way to get softly modulating sounds. However, we have seen with HILITE TRACE that a remarkable number of windows can go before the recognisability of the sound is seriously affected, so there is plenty of scope to soften the edges of a sound before recognisability is lost.

A high *blur* factor combined with a low *trace* factor greatly reduces and smooths the sound: reduces it to a gentle murmur. You might also consider using FOCUS ACCU to alter/intensify the blurring effect.

Also see **BLUR BLUR**, **HILITE TRACE** and **FOCUS ACCU**

End of HILITE BLTR

HILITE FILTER – Filter the spectrum

Usage

hilite filter 1–4 *infile outfile frq1 Q*
hilite filter 5–6 *infile outfile frq1 Q gain*
hilite filter 7–10 *infile outfile frq1 frq2 Q*
hilite filter 11–12 *infile outfile frq1 frq2 Q gain*

Modes

1 high pass filter
2 high pass filter (normalised output)
3 low pass filter
4 low filter (normalised output)
5 high pass filter with gain
6 low pass filter with gain
7 band pass filter
8 band pass filter (normalised output)
9 notch filter
10 notch filter (normalised output)
11 band pass filter with gain
12 notch filter with gain

Parameters

infile – input analysis file made with PVOC
outfile – output analysis file
frq1 – filter cutoff frequency
frq1 with *frq2* – limits of filter band
Q – width of filter skirts, in Hz (Range: > 0)

Here one is entering filter bandwidth directly. The smaller the Hz value for *Q* (i.e., the filter's bandwidth), the narrower the pitch focus of the filter.

gain – amplification of the resulting sound

frq1, *frq2* and *Q* may vary over time

Understanding the HILITE FILTER Process

These are standard filtering options, which can be applied directly in the spectral domain. Note that *Q* is entered as a frequency bandwidth, which hopefully will be a more intuitive way to use this parameter.

Musical Applications

The key facilities here are the time-varying frequency settings and Q. One can move the frequency placement about as the sound progresses, as well as change the sharpness (pitch focus) of the filter. Suppose, for example, you extracted a pitch trace and saved the data as a breakpoint file. This file could be used as a time-varying *frq1* with a time-varying *Q*, so that the salient pitch content of the file can be made to fade in and out. Similarly, upward and downward sweeps which begin with a broad and end with a sharp pitch focus or *v.vs.* (*Q*), affecting changes over time, emerging or disappearing pitches etc.

The following programs provide data which could be useful when deciding how to set the filter frequencies (use BACK in your Browser to return here to HILITE FILTER):

SPECINFO PEAK and **SPECINFO REPORT** provide information on (time-varying) peaks in the spectral envelope. **REPITCH GETPITCH** can extract a pitch trace, saving it as either a binary pitch data file or a breakpoint file, and **PITCHINFO WRITE** can convert a binary pitch data file into a breakpoint file. **PITCHINFO INFO** gives an overview of the (main) pitch content of a binary pitch data file.

End of HILITE FILTER

GLISTEN – Randomly partition the spectrum into bins and play back in order

Usage

glisten glisten *inanalfile outanalfile grpdiv setdur* [**-p***pitchshift*] [**-d***durrand*] [**-v***divrand*]

Example command line to enliven an otherwise fairly static spectrum:

```
glisten glisten in.ana out.ana 4 250 -p3 -d0.7 -v0.25
```

Parameters

inanalfile – input analysis file
outanalfile – output analysis file
grpdiv – the number of sets into which to divide the analysis-channels. For example, when *grpdiv* = 4, the program will process the partitions and channels randomly among 4 sets with the total number of channels ÷ 4 per set. **Note that *grpdiv* must be an exact divisor of the channel count.** (Range: 2 to channel-count)
setdur – the number of windows for which a set-of-channels persists before we switch to the next set-of-chans. (Range: 1 to 1024)
-p*pitchshift* – The maximum range in (possibly fractional) semitones of random plus or minus pitch shifting (i.e., upwards or downwards) of each channel set. (Range: 0.0 to 12.0. Default: 0.0)
-d*durrand* – the randomisation of *setdur* between 1 and *setdur*. (Range: 0 to 1)
-v*divrand* – randomise the number of channels in each set in a group. (Range: 0 to 1.)
In the normal case, each set-of-channels has an equal number of channels. When *divrand* is > 0, a group will have sets of different sizes.

Understanding the GLISTEN Process

All the channels of the analysis are partitioned into *N* mutually exclusive sets, with channels assigned at random to each set. These sets form a complete group. Then channels in the first set are played, for *setdur* windows, at which point the channels in the 2nd set are played for *setdur* windows and so on.

As this happens, we are progressing through the original file at its original rate, the process determining merely which channels are used – the others being zeroed. Once all *N* sets are exhausted, a new group is made by random partition, and so on.

Musical Applications

This process is meant to 'animate' a fixed or relatively static spectrum. The channels of the spectrum are selected (in groups) at random and played back in sequence.

End of GLISTEN

HILITE GREQ – Graphic EQ type filter on the spectrum

Usage

hilite greq mode *infile outfile filtfile* **-r**

Modes

1 single bandwidth for all filter bands

filtfile has ONE bandwidth. This is expressed in octaves or proportions of octaves (e.g., 1 or 0.5), and this value is followed by the centre frequencies in Hz of all the filter bands of the graphic equaliser. The file is a list of numbers, so it doesn't matter whether you use newlines, spaces etc., as long as the numbers are distinct.
Single bandwidth *filtfile* example

0.75 40 80 160 320 640

2 separate bandwidths for each filter band

filtfile has a pair of values defining each filter band. These are the bandwidth in octaves (expressed as in Mode 1) and the centre frequency of the band.
Separate bandwidths *filtfile* example

1.0 40

0.5 80

2.6 240

Parameters

infile – input analysis file made with PVOC
outfile – output analysis file
filtfile – text file defining the graphic equaliser (as illustrated above)
-r – Band reject (notch) filter. Default is a bandpass filter.

Understanding the HILITE GREQ Process

This is in concept a standard graphic equaliser, with a choice between bands of equal size or bands of varyingly specified sizes. A sonogram display may help to show how strong signal areas may be reduced (or enhanced by removing bands of frequencies which lie between them). Also see the spectral information gathering functions listed below.

Musical Applications

The graphic equaliser is typically used to boost certain selected parts of a sound, and reduce others, such as high level hiss or low booming. This 'EQ type' filter does not have a gain parameter as such and focuses on highlighting (or removing) the selected bands.

The following programs provide data which could be useful when deciding how to set the filter frequencies (use BACK in your Browser to return here to HILITE FILTER):

SPECINFO PEAK and **SPECINFO REPORT** provide information on (time-varying) peaks in the spectral envelope. **REPITCH GETPITCH** can extract a pitch trace, saving it as either a binary pitch data file or a breakpoint file, and **PITCHINFO CONVERT** can convert a binary pitch data file into a breakpoint file. **PITCHINFO INFO** gives an overview of the (main) pitch content of a binary pitch data file.

End of HILITE GREQ

HILITE PLUCK – Emphasise spectral changes (use e.g., with HILITE ARPEG)

Usage

hilite pluck *infile outfile gain*

Parameters

infile – input analysis file made with PVOC
outfile – output analysis file
gain – amplitude gain applied to newly prominent spectral components (Range: 1 to 1000)

gain may vary over time

Understanding the HILITE PLUCK Process

In analysis files generated from complex soundfiles by functions such as HILITE TRACE and HILITE ARPEG, new partials will enter (and leave) the spectrum, window by window, creating a kind of 'resultant melody'. HILITE PLUCK emphasises the entry of these new partials by boosting their amplitude at the point of entry. If the amplitude boost is large, such as increased by a *gain* of 10 or more, the entries may suggest a 'plucking' of the entering partials.

Note that the 'pluck' effect is added to the onset of partials. This is quite different from emphasising the attack transient of a sound in the amplitude envelope. See **ENVEL PLUCK** for this, where a noise component is introduced into the attack transient of a sound.

Musical Applications

This procedure will introduce into the sound a somewhat random rhythm of short pulses. These pulses will attract the ear to the entering partials and whatever harmonic relationship they may have such as the harmonic partials retained after running SPEC BARE. SPEC BARE + HILITE TRACE + HILITE PLUCK form an interesting combination of functions to play with, and HILITE PLUCK can also be used to bring out the partials which HILITE ARPEG makes more audible.

Also see: **HILITE ARPEG**, **HILITE TRACE** and **SPEC BARE**

End of HILITE PLUCK

HILITE TRACE – Highlight *n* loudest partials, at each moment (window) in time

Usage

hilite trace 1 *infile outfile N*
hilite trace 2 *infile outfile N lofrq* [**-r**]
hilite trace 3 *infile outfile N hifrq* [**-r**]
hilite trace 4 *infile outfile N lofrq hifrq* [**-r**]

Modes

1 Select loudest spectral components
2 Select loudest from above *lofrq*: reject all spectral data below *lofrq*
3 Select loudest from below *hifrq*: reject all spectral data above *hifrq*
4 Select loudest from between *lofrq* and *hifrq*: reject data outside

Parameters

infile – input analysis file made with PVOC
outfile – output analysis file
N – the trace index: the number of spectral components to retain
lofreq – frequency below which spectral data is rejected
hifreq – frequency above which spectral data is rejected
-r – If trace index *N* is greater than the number of channels in the filterband (defined by *lofrq* & *hifrq*):

RETAIN the loudest channels OUTSIDE the filterband
(Default: always omit channels outside the filterband)

N, *lofrq* and *hifrq* may vary over time

Understanding the HILITE TRACE Process

HILITE TRACE looks for and retains only the *N* loudest partials in the analysis data on a window-by-window basis. This reduces the data in the spectral dimension and produces an aural 'trace' of the original sound.

With non-'noisy' sources it is necessary to reduce the number of channels quite considerably to make any appreciable aural change to the source sound. Even the 10 loudest channels will retain a surprising amount of the original sound. The flip-side of this is that HILITE TRACE can be used to 'clean' a sound, if a certain amount of data loss is not a problem (see below).

My understanding of this procedure is that it deals with analysis channels, not directly with partials. It will retain whichever partial is contained in a given analysis channel window at any given moment in time, and this is likely to change. To find specific partials requires 'peak tracking', and this is not something which the CDP software is able to do. Thus a sonogram of 1 channel retained by HILITE TRACE will probably show a number of different frequencies. The frequency width of an analysis channel is obtained by dividing the sample rate by the number of channels: e.g., `44100 / 1024 = 43`. This enables you to calculate the width of the frequency band

that a given number of channels will cover. Remember that partials are more spread out in higher sounds (frequency is logarithmic), so low sounds are likely to contain more partials within a smaller band, while the same width might find relatively few in a high sound. For an alternative approach, see http://mpex.prosoniq.com.[AE]

The HILITE TRACE filtering effect can be further controlled by using the *lofrq* and *hifrq* parameters. The default is to **omit** the data outside the selected area, but if the **-r** flag is used, this data is **retained**.

For example, the frequency band can be focused into a narrower band by creating a breakpoint file for *hifrq* which starts high and moves lower, and creating a breakpoint file for *lofrq* which starts low and moves higher (but stays below the final frequency value in *hifrq*).

Musical Applications

Initially, this process can be used to clean up some of the ambient noise which is part of a sound source, if for example, only 50 or 100 channels are retained. However, some spectral information may be lost as well, when cleanup is done in this way.

However, as one moves down to 50, 40, 30, 20, 10, 5 ..., less and less of the original remains, though it is surprising how low one has to go to get a really 'ghostly' result. At this stage, the original is being used simply as a source for another sound. Strongly and rapidly changing sounds can produce a burbling effect.

Also see: **BLUR SUPPRESS**, **HILITE BLTR** and **SPEC CLEAN**

End of HILITE TRACE

HILITE VOWELS – Impose vowels on a sound

Usage

hilite vowels *infile outfile vowelfile halfwidth steepness range threshold*

Parameters

infile – input analysis file onto which to impose the vowels
outfile – resultant analysis file
vowelfile – a text file containing paired *times* (must start at 0 and increase) and *vowels*, where vowels can be:

VOWEL	AS IN	VOWEL	AS IN	VOWEL	AS IN	VOWEL	AS IN
ee	*heat*	**i**	*hit*	**e**	*pet*	**ai**	*hate*
a	*pat*	**ar**	*heart*	**o**	*hot*	**or**	*taught*
oa	*boat*	**u**	*hood*	**oo**	*mood*		
xx	Southern English *hub*			**x**	neutral vowel in *herb* or *the*		

halfwidth – half-width of formant peaks in Hz as a fraction of the formant centre frequency. Default: 0.25
steepness – steepness of formant peaks, similar to 'Q'in filtering: steeper means more resonant. Range: 0.1 to 10. Default: 3
range – the formant peaks stand above the signal floor. The *range* of the formant peaks is therefore a part of the total range of the signal. *Range* is a ratio of (maximum) peak height to (maximum) total range. Default: 0.95
threshold – the spectral window's level is compared with the vowel envelope level. If it exceeds a certain proportion of that level, it is forced to the vowel envelope level. Otherwise, it remains where it is lest it amplify background noise artificially. *Threshold* defines this proportion. Default: 0.5

Understanding the HILITE VOWELS Process

The *vowelfile* is a straightforward breakpoint file, with the *times* in the left column and the *vowels* as defined above in the right column:

```
[times  vowels]
 0.0    ai
 3.0    ee
 6.0    oa
 9.0    ar
12.0    i
```

This process allows a sequence of vowels to be imposed over a spectrum derived from an input sound. The spectrum itself must have enough energy over all frequency ranges for the vowel formants to 'bite' e.g., a very pure pitch sound (a sine tone) only has energy at one place in the spectrum. The vowel formant

has nothing much to shape, and the vowel sound will not be formed. Noisy sounds are best. If you want to impose vowel formants on strongly pitched sounds if may be best to extract the pitch and then use REPITCH VOWELS OVER PITCHFILE.

The process glides between vowels specified at different times in a breakpont file. e.g. if you set the vowel 'a' at time 0 and also at time '1', the output will generate an 'a' sound. But if the vowel at time 1 is set to 'i', the output will do a vowel-glide (diphthong) from 'a' to 'i'. Speech characteristics can thus be imposed on pitch contours derived from pitched source.

The sound onto which the vowels are imposed needs to a rich, noisy sound with fairly evenly spread energy. Otherwise, the vowel contours may find that they have nothing in the original sound to with which to work.

Musical Applications

This process can be used to create time-varying spectra or even the illusion of speech.

End of HILITE VOWELS

Technical Discussion about spectral amplitude

See **spectral envelope** and **formants** in the Technical Glossary.

End of Technical Discussion

Last Updated 18 Sep. 2015 -- HTML5 version
Documentation: Archer Endrich, revised R.Fraser