lifter vst

cepstral impulse resynthesis build 2 by xoxos

what

The cepstrum divides resonances from source signals, described as "deconvolution of the source and filter". It is used to create impulses containing spectral information. Triggering these impulses at audio rate creates oscillators with the spectral contour from the cepstral analysis.

The limitation of this method is that it produces the harmonic series. It is useful for emulating instruments with the series of harmonic partials, but not for instruments with inharmonic or absent partials, like bells or clarinet. The advantages of impulse resynthesis are efficiency and immediacy (zero latency, every specification is instantaneous).

After building my initial cepstral impulse synth, kaze, I realised that one could generate a second set of impulses with a 90° phase offset, allowing for complex (2 dimensional) processing, such as frequency shifting, so that the partials could be expanded or compressed. Predictably, because of the influence of the fundamental pitch, a pure frequency shift is not possible, as a second set of partials are reflected around the fundamental. But the ability to produce inharmonic partials significantly improves the utility of the technique.

As kaze, lifter creates pitched and unpitched oscillators from the cepstral filter data, achieving latency free, efficient resynthesis with continuous, recallable, instantaneous specification of pitch, phase, spectral shift and pitched/noise signal balance. Lifter adds side band modulation to this list.

how

Cepstral vernacular reverses the first part of signal processing terms, eg. quefrency, liftering and alanysis. The lifter parameter specifies the separation of filter and source. Lower values have less spectral resolution, higher values start to bias the resulting spectral contour with the source.

Spectra of the wavefile are taken at 128 periodic points, producing better resolution with shorter files. The pitched oscillator crossfades between antialiased impulses. The unpitched oscillator uses tables of 16384 samples, which would repeat about 2.69 times a second at 44.1kHz with no spectral shift. This may produce overt repeating but its very computationally efficient.

Preparation of each frame requires four or five fourier operations, depending on if noise is opted. A total of 128 frames means some cpu lock up is expected during resynthesis (some part of a second on my single core). This occurs when the lifter parameter is moved (eg. don't drag it slowly), and when the wavefile is loaded, which takes a while.

sidebands

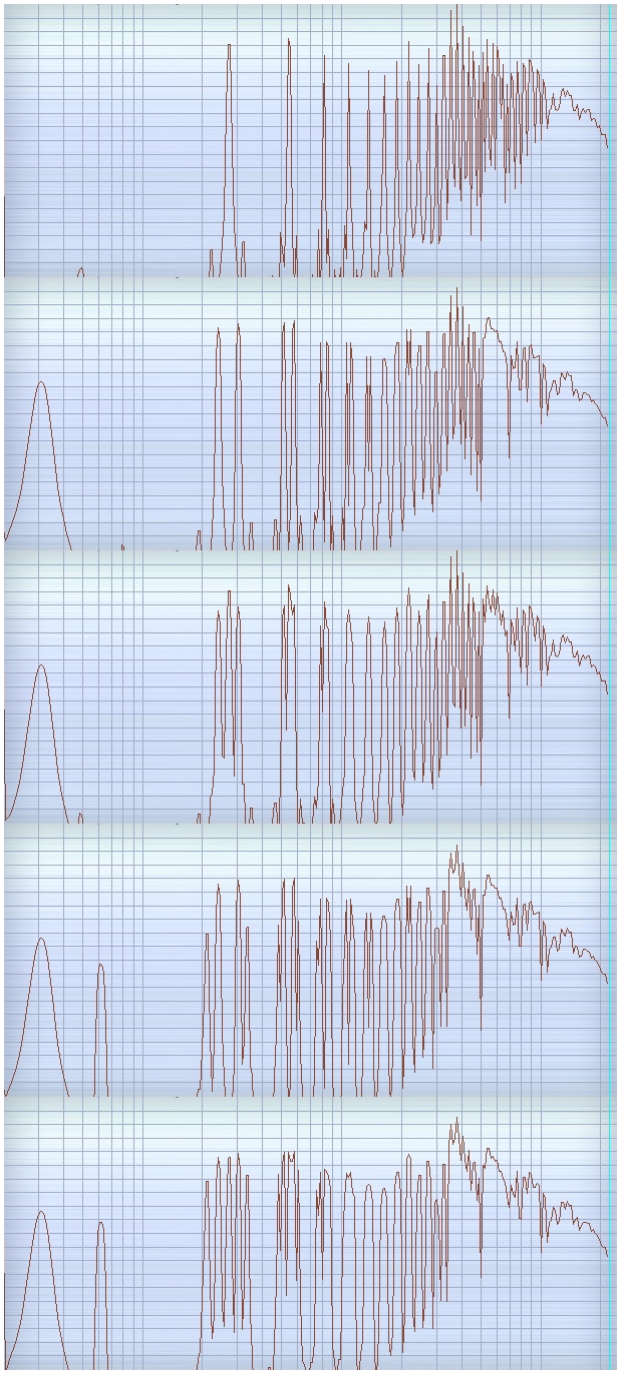
Sidebands are generated as a function of source frequency plus modulation frequency. A fixed modulation frequency produces the same rate of harmonic beating or phasing regardless of the source frequency.

Even numbered sideband settings suppress the source frequency. Different modes are useful in emulation of different acoustic systems. Two "2 band" options select emphasis of the upper or lower band, reflecting upward and downward shifts reflected around the fundamental.

scale

I implemented a "correction" function that would retune the oscillator to the lower or upper sideband. The result of this was that changing the sideband frequency did achieve an expansion or compression of the partials, of course still generating the extra set of partials. My "fundamental correction" method had an unforeseen side effect in that it also rescaled the spectral contour as well as the partials.

The result of this is that the gui scale function, which appears to be a single button below the sideband frequency control, actually pages between three modes - no correction, "expand," and "contract," depending on whether the lower or upper sideband is used. There is no label or gui readout for which mode it is in, but in use, the sideband frequency will have a dreamatically diffferent effect. It mak teh interesting sound.



help i don't know

There are no presets because wavefiles take a very long time to load for some reason (about the same length as the sample). After the sample is loaded, the oscillator will play the frame at the read position. If this position in the file is silent, no sound will be produced.

In order to play the file through conventionally, a linear modulator must be applied to the read parameter. The easiest way to do this is to use a linear lfo (contour 0). Set gate sync for the lfo on, so it will play in the same place when you hit a key, and use a mod assign to send it to the read parameter. Change the speed of the lfo to change the playback rate. Change the lfo phase param to change the starting position. Chances are you're listening to it playing backwards, so invert the mod assign.

This synth was developed very quickly to make it available in some form and allow me to do other things. The basic method can perform well for some emulative applications (which is why the simpler build, kaze, is still useful). Some parts are unrefined in comparison to the expectations of users accustomed to commercial development, eg. as consonants are very brief, they will lose resolution with longer files since frames are analysed at 128 equally spaced points. Prepping wavefiles may be worthwhile. The analysis for pitched versus unpitched signal components is also very simple, so in use, balance of noise versus oscillator may need to be "worked with". Ultimately this thing can throw out a lot of dynamic timbre.

The use of complex signals also avails stereo output. With lower sideband settings, this will produce a rotating effect. If it is too cyclic, use a random lfo to very subtly modulate the sideband frequency, very nice. Very small sideband settings can "warm" or "naturalise" integer harmonics, to sound less vocodery.