ELEC-C5340
Applied Signal Processing
2016

Vocoder as an Effect

Vocoder is a system which is originally created for telecommunications as a tool for coding speech signal. Essentially what it does is that it analyzes input signal with a certain method and resulting data can be transmitted. At the receiving end this data is again synthesized for an output. However, since its creation, vocoder has also been (mis-)used as a musical effect.[1] Also it has been used in science fiction movies and series quite widely.

In this paper we describe kind of vocoder as an effect which is based to the original analog vocoder design using filter banks. Vocoder can also be implemented based on e.g. linear prediction, that is also a feasible choose, but not described here.

Realization

Implementing a filter bank based vocoder digitally is quite a straightforward task but requires some planning. This is because there are a multitude of parallel signal paths and the data has to be managed carefully so that everything works correctly. In this assignment the vocoder is done in time domain.

The block diagram of the vocoder can be seen in figure 1. Starting from the left, there are two input signals. Control signal is usually speech and carrier signal can be almost anything but should have wide-band spectrum. Both signals are then filtered through same filter bank.[2] Then the filtered control signal is analyzed by estimating the energy at the frequency band. As a result a gain term for controlling carrier signal is created. For achieving more constant results it is usually prudent to apply some form on normalization at this point to the gain terms or the estimated energy. Applying normalization then again usually needs some form of noise gate to be applied to reduce unwanted noise. After this the gain values calculated from the control signal frequency bands are multiplied

[1] For example Kraftwerk and especially Daft Punk.

[2] Actually for musical usage it is not necessary to have exactly same filter bank for both signals. It is only done here to make the work a bit simpler.

with corresponding carrier signal bands. Now the carrier frequency bands can be added back together to produce the output signal.

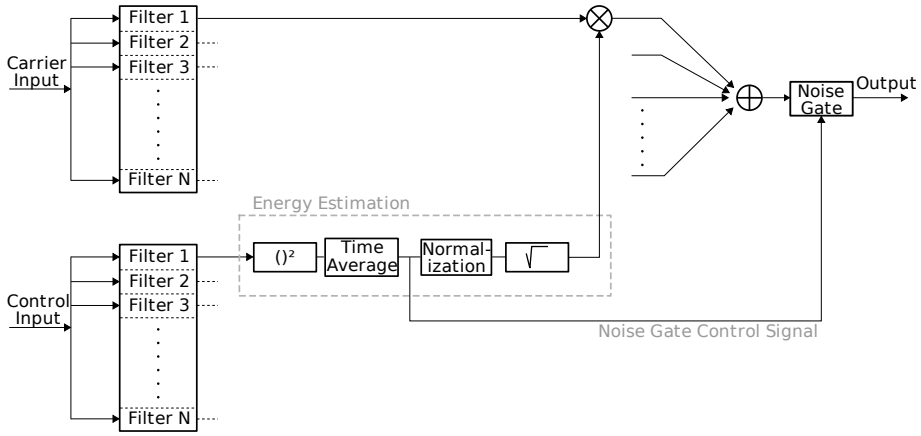

Figure 1: Block diagram of the vocoder

As it can be noted from from the block diagram, vocoder in a sense applies a time-changing filter to a signal. Thus the result of vocoder is a signal which has carrier signal's harmonic structure and spectral content which is modified with the spectral envelope of the control signal. Depending on the input signal the result can be talking synthesizer or something totally different. More of general information about vocoders can be found in Wikipedia and in this archived message by Harnden. Also a software version of a vocoder effect can be downloaded from website by Borsboom.

Filter banks

Filter banks in this assignment are implemented with a Gammatone filter bank. It is widely used to approximate the frequency resolution of the human hearing thus is a good choice for dividing the signal to frequency bands. This filter band is not perfect reconstruction but is reasonably close for musical purposes. Eight order filter bank can be created using the supplied MATLAB script (course material). Note that this script actually creates a cascade of four second order filters instead of an eight order filter. This is because eight order filters would be unstable due to round-off errors.

Energy estimation

Energy estimation inside one frequency band is done simply by squaring the input gain value and then time averaging the result. Time averaging can be done with any suitable method but there are two quite simple and good solutions.

The first solution is to use exponentially decaying time window. Using it implements a "leaking integrator". Exponentially decaying window in in this case is implemented using the formula in equation 1. Correct constant a can be calculated from the time constant τ of the desired filter and sampling frequency f_s using equation 2. This is equal to a first order low-pass one pole IIR filter. Correct value for constant a can be found through trial and error. A good start value for iteration is about 100 milliseconds and the correct value is the one which makes the vocoder sound good.

$$E(n) = (1-a)\,E(n-1) + a\,[x(n)]^2 \tag{1}$$

$$a = \frac{1}{\tau f_s} \tag{2}$$

The second solution is to calculate average with a window function. Rectangular window is simple to implement as a FIR filter where each coefficient is at same value. However, there is an equal version of the same filter using an alternative, more efficient, form with a difference equation seen in equation 3. Here N is the length of the window as samples. Correct length should be found again through trial and error.

$$E(n) = E(n-1) + \frac{1}{N}\,[x(n)]^2 - \frac{1}{N}\,[x(n-N)]^2 \tag{3}$$

This window function corresponds to a transfer function in equation 4. $\frac{1}{N}$ is the gain term of the filter and ensures that the signal is not amplified in the process.

$$H(z) = \frac{1}{N}\frac{1-z^{-N}}{1-z^{-1}} \tag{4}$$

Different time averaging methods give slightly different sounding results. It should be noted that the result of averaging should always be positive as there is no such thing as negative energy in the real world.

Finally the gain value is obtained by taking square root of the estimated energy. ~~Some processors have square root algorithms already implemented and on the~~

~~others, an iterative or approximative can be used. Consult assistants for more information.~~

Normalization and noise gate

Normalization and noise gate is not actually needed to do in this assignment but after few years it has been noted that implementing them makes it much easier to actually get the vocoder sound good.

Normalization should effect the gain values calculated from control signal so that it does not matter that much how loud the signal is. There are few choices for the algorithm which give different results. Simplest method is to find at each time instant the largest gain value between frequency bands. All gain values are then normalized by dividing with that value. This method normalizes largest gain value always to one. This method can be seen in equation 5.

$$g_i(n) = \frac{\sqrt{E_i(n)}}{\max_{j=1}^{N} \left\{ \sqrt{E_j(n)} \right\}} \tag{5}$$

A bit more complex approach is to normalize the loudness of the signal. Quite good approximation is to divide each gain value before taking the square root with total signal energy. This is as per equation 6. This method gives more constant loudness value for this

$$g_i(n) = \sqrt{\frac{E_i(n)}{\sum_{j=1}^{N} E_j(n)}} \tag{6}$$

Both of these methods do the work well but again the results will have some differences.

The downside of normalization is that it also boosts background noise during silent input. This unwanted side effect can be prevented to some extent by implementing a noise gate.

Noise gate is a gate function used for changing sound to silent when the input is at low energy. Generally this is done using a decision threshold level where all values which are under the threshold will be put to zero. Values over the threshold are left untouched. The function can be seen in figure 2.

This noise gate should be applied at correct position. Most efficient method is to apply the gate at the end of the signal path as then there only needs to be one operation. However, the state of the gate should be calculated from the

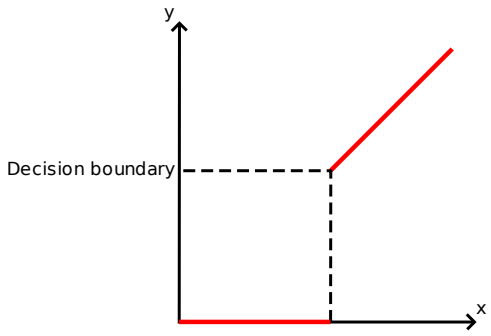

Figure 2: Gate function with the decision threshold shown.

energy data of the control signal before normalization. A very good data to use is the total energy of all bands which is already part of equation 6.

This noise gate works well otherwise but rapid changes to zero create unwanted nonlinearities which sound really awful. These can be removed with some creative slowing of the change. One choice is to low pass filter rapid changes to slower ones. Probably a better one is to simply slow the change using a couple of if-conditionals.

Realtime implementation

You are advised to design a working vocoder effect in Matlab. Then you can implement your design as a VST plug-in or an audio application.

Enhancements like utilizing a noise gate and normalization are a plus.

References

Emanuel Borsboom. Vocoder software http://www.epiphyte.ca/code/vocoder.html.

SPÄNK course material. MATLAB script for creating gammatone filters.

Eric Harnden. General explanation about vocoder http://riksun.riken.go.jp/archives/misc/MIDI/DOC/vocoder.txt.