VoiceOfFaust

Bart Brouns
studio magnetophon
Biesenwal 3
Maastricht, Netherlands, 6211 AD
bart@magnetophon.nl

Abstract

VoiceOfFaust turns any monophonic sound into a synthesizer, preserving the pitch and spectral dynamics of the input.

There are 7 synthesizer and two effect algorithms:

- a classic channel vocoder
- a couple of vocoders based on oscillators with controllable formants:
  - CZ resonant oscillators
  - PAF oscillators
  - FM oscillators
  - FOF oscillators
- FM with modulation by the voice
- ring-modulation
- Karplus-Strong used as an effect
- Phase modulation used as an effect

Keywords

Synthesis, Signal Processing, Audio Plugins.

1 Introduction

VoiceOfFaust turns any monophonic sound into a synthesizer, preserving the pitch and spectral dynamics of the input. It is written in Faust [1], and uses a pitch tracker in Pure Data [2].

It consists of:

- an external pitch tracker: helmholtz~ [3] by Katja Vetter.
- a compressor/expander, called qompander [4], ported to Faust.

There are 7 synthesizer and two effect algorithms:

- a classic channel vocoder
- a couple of vocoders based on oscillators with controllable formants:
  - CZ resonant oscillators
  - PAF oscillators
  - FM oscillators
  - FOF oscillators
- FM with modulation by the voice
- ring-modulation
- Karplus-Strong used as an effect
- Phase modulation used as an effect

The features include:

- all oscillators are synchronized to a single saw-wave, so they stay in phase, unless you don't want them to
- powerful parameter mapping system lets you set different parameter values for each band, without having to set them all separately
- formant compression/expansion: Make the output spectrum more flat or more resonant, at the twist of a knob.
- flexible in and output routing: change the character of the synth.
- all parameters, including routing, but except the octave, are step-less, meaning any 'preset' can morph into any other.
- multi-band deEsser and reEsser
- optionally use as a master-slave pair:
  The master is a saw-oscilator driven by the (external) pitchtacker, and the slaves contain everything else, synced to the master.
  This makes it possible to run the slaves as plugins.
- configuration file:
  Through this file, lot's of options can be set at compile time, allowing you to adapt the synth to the amount of CPU power and screen real-estate available.
  Some of the highlights:
- number of bands of the vocoders
- number of output channels
- whether we want ambisonics output
- whether a vocoder has one set of oscillators, or a separate set of oscillators per output.

2 Vocoders

2.1 Common features of all vocoders

2.1.1 Parameter mapping system

The parameters for the vocoders use a very flexible control system:

Each parameter has a bottom and a top knob, where the bottom changes the value at the lowest formant band, and the top the value at the highest formant band.

The rest of the formant bands get values that are evenly spaced in between.

For some of them that means linear spacing, for others logarithmic spacing.

For even more flexibility there is a parametric mid:

You set it's value and band number and the parameter values are now:

- 'bottom' at the lowest band, going to:
- 'mid value' at band nr 'mid band', going to:
- 'top value' at the highest band.

Kind of like a parametric mid in equalizers.

If that's all a bit too much, just set ``para`` to 0 in the configuration file, and you'll have just the top and bottom settings.

2.1.2 Formant compression/expansion

Scale the volume of each band relative to the others:

- 0 = all bands at average volume
- 1 = normal
- 2 = expansion

expansion here means:

- the loudest band stays the same
- soft bands get softer

Because low frequencies contain more energy than high ones, a lot of expansion will make your sound duller.
To counteract that, you can apply a weighting filter, settable from

- 0 = no weighting
- 1 = A-weighting
- 2 = ITU-R 468 weighting

2.1.3 DeEsser

To tame harsh esses, especially when using some formant compression/expansion, there is a deEsser:
It has all the usual controls, but since we already are working with signals that are split up in bands, with known volumes,
it was implemented rather differently:

- multiband, yet much cheaper,
- without additional filters, even for the sidechain,
- and with a dB per octave knob for the sidechain, from 0dB/oct (bypass), to 60dB/oct (fully ignore the lows).

It also has a (badly named) noise strenght parameter: it uses the fidelity parameter from the external pitchtracker to judge if a sound is an S.
When you turn it up, the deEsser gets disabled when the pitchtracker claims a sound is pitched.
See [3] for more info.

2.1.4 ReEsser

Disabled by default, but can be enabled in the configuration file.
It replaces or augments the reduced highs caused by the deEsser.

2.1.5 DoubleOscs

This is a compile option, with two settings:

- 0 = have one oscillators for each formant frequency
- 1 = creates a separate set of oscillators for each output channel, with their phase modulations reversed.

2.1.6 In and output routing

The vocoders can mix their bands together in various ways:
We can send all the low bands left and the high ones right, we can alternate the bands between left and right, we can do various mid-side variations
we can even do a full Hadamard matrix.
All of these, and more, can be cross-faded between.
In the classicVocoder, a similar routing matrix sits between the oscillators and the filters.

2.1.7 Phase parameters

Since all[1] formants are made by separate oscillators that are synced to a single master oscillator, you can set their phases relative to each other.
This allows them to sound like one oscillator when they have static phase relationships, and to sound like many detuned oscillators when their phases are moving.

[1] except for the classicVocoder.

Together with the output routing, it can also create interesting cancellation effects.
For example, with the default settings for the FMvocoder, the formants are one octave up from where you'd expect them to be.
When you change the phase or the output routing, they drop down.

These settings are available:

- static phases
- amount of modulation by low pass filtered noise
- the cutoff frequency of the noise filters

2.2 Features of individual vocoders

2.2.1 ClassicVocoder

Block-diagram:
https://magnetophon.github.io/VoiceOfFaust/images/classicVocoder-svg/process.svg

A classic channel vocoder, with:

- a "super-saw" that can be cross-faded to a “super-pulse", free after Adam Szabo [5].

* flexible Q and frequency setting for the filters
* an elaborate feedback and distortion matrix around the filters

The gui of the classicVocoder has two sections:
First oscillators, containing the parameters for the carrier oscillators.
These are regular virtual analog oscillators, with the following parameters:

- cross-fade between oscillators and noise
- cross-fade between sawtooth and pulse wave
- width of the pulse wave
- mix between a single oscilators and multiple detuned ones
- detuning amount

Second filters, containing the parameters for the synthesis filters:

- bottom, mid and top set the resonant frequencies
- Q for bandwidth
- a feedback matrix. each filter gets fed back a variable amount of:
  - itself
  - it's higher neighbor
  - it's lower neighbor
  - all other filters
  - distortion amount
  - DC offset

2.2.2 CZvocoder

Block-diagram:
https://magnetophon.github.io/VoiceOfFaust/images/czVocoder-svg/process.svg

This is the simplest of the vocoders made out of formant oscilators.
The oscillators where ported from a pd patch by Mike Moser-Booth [6].

You can adjust:

- the formant frequencies
- the phase parameters

2.2.3 PAFvocoder

Block-diagram:
https://magnetophon.github.io/VoiceOfFaust/images/PAFvocoder-svg/process.svg

The oscillators where ported from a pd patch by Miller Puckette [7].

It also has frequencies and phases, but adds index for brightness.

2.2.4 FMvocoder

Block-diagram:
https://magnetophon.github.io/VoiceOfFaust/images/FMvocoder-svg/process.svg

The oscillators where based on code by Chris Chafe [8].

Same parameters, different sound.

2.2.5 FOFvocoder

Block-diagram:
https://magnetophon.github.io/VoiceOfFaust/images/FOFvocoder-svg/process.svg

Original idea by Xavier Rodet [9].
based on code by Michael Jørgen Olsen [10].
Also has frequencies and phases, but adds:

- skirt and decay:
  Two settings that influence the brightness of each band
- Octavation index
  Normally zero. If greater than zero, lowers the effective frequency by attenuating odd-numbered sinebursts.

Whole numbers are full octaves, fractions transitional.
Inspired by an algorithm in Csound [11].

3 Other synthesizers

These are all synths that are not based on vocoders.

3.1 Features of individual synths

3.1.1 FMsinger

Block-diagram:
https://magnetophon.github.io/VoiceOfFaust/images/FMsinger-svg/process.svg

A sine wave that modulates its frequency with the input signal.
There are five of these, one per octave, and each one has:

- volume
- modulation index
- modulation dynamics
  This fades between 3 settings:
  - no dynamics: the amount of modulation stays constant with varying input signal
  - normal dynamics: more input volume equals more modulation
  - inverted dynamics: more input equals less modulation.

3.1.2 CZringmod

Block-diagram:
https://magnetophon.github.io/VoiceOfFaust/images/CZringmod-svg/process.svg

Ringmodulates the input audio with emulations of Casio CZ oscillators.
Again five octaves, with each octave containing three different oscillators:

- square and pulse, each having volume and index (brightness) controls
- reso, having a volume and a resonance multiplier:
  This is a formant oscillator, and it's resonant frequency is multiplied by the formant setting top right.
  It is intended to be used with an external formant tracker.
- There is a global width parameter that controls a delay on the oscillators for one output.
  The delay time is relative to the frequency.
  Because this delay is applied to just the oscillators, and before the ringmodulation, the sound of both output channels arrives simultaneously.
  This creates a mono-compatible widening of the stereo image.

3.1.3 KarplusStrongSinger

Block-diagram:
https://magnetophon.github.io/VoiceOfFaust/images/KarplusStrongSinger-svg/process.svg

This takes the idea of a Karplus Strong algorithm [12], but instead of noise, it uses the input signal.
The feedback is ran trough an allpass filter, modulated with an LFO; adapted from the nonLinearModulator in instrument.lib.
To keep the level from going out of control, there is a limiter in the feedback path.
Parallel to the delay is a separate nonLinearModulator.
Globally you can set:

- octave
- output volume
- threshold of the limiter

For the allpass filters you can set:

- amount of phase shift
- difference in phase shift between left and right (yeah, I lied, there are two of everything)
- amount of modulation by the LFO
- frequency of the LFO, relative to the main pitch
- phase offset between the left and right LFO's.

To round things off there is a volume for the dry path and a feedback amount for the delayed one.

3.1.4 KarplusStrongSingerMaxi

Block-diagram:
https://magnetophon.github.io/VoiceOfFaust/images/KarplusStrongSingerMaxi-svg/process.svg

To have more voice control of the spectrum, this one has a kind of vocoder in the feedback path.
Since we don't want the average volume of the feedback path changing much, only the volumes relative to the other bands, the vocoder is made out of equalizers, not bandpass filters.
You can adjust it's

- strength: from bypass to 'fully equalized'

- cut/boost; steplessly vary between
  - -1 = all bands have negative gain, except the strongest, which is at 0
  - 0 = the average gain of the bands is 0.
  - +1 = the all bands have positive gain, except the weakest, which is at 0
- top and bottom frequencies
- Q factor

4 Master-slave

This is a workaround for the need for an external pitchtracker, making it possible to use the synths and effects as plugins.
It has the nice side effect that your sounds become fully deterministic:
because a pitchtracker will always output slightly different data, or at least at slightly different moments relative to the audio, the output audio can sometimes change quite a bit from run to run.
The master is a small program that receives the audio and the OSC messages from the external pitch tracker, and outputs:

- a copy of the input audio
- a saw wave defining the pitch and phase
- the value of fidelity, from the pitch tracker, as audio.

The slaves are synths and effects that input the above three signals.
The outputs of the master can be recorded into a looper or DAW, and be used as song building blocks, without needing the pitch tracker.
This makes it possible to switch synths, automate parameters, etc.

5 Strengths and weaknessesses of Faust

The Faust language has some big advantages.
The common perks of the language apply. For me, the biggest ones are:

- Quick implementation of ideas.
- If it sounds right, it is right. There won't be any crashes, memory leaks or other bugs.
- Write once, deploy everywhere.
- The block diagrams help with debugging and documentation.
- Fast running code.
- Automatic parrallelisation.

In this project it was also very helpful to be able to easily parameterize things like the number of bands. Related: the input and output routing wouldn't be nearly as easy and fun to implement in most languages, as they lean heavily on Fausts splitting and combinatory operators.

Since this idea has been implemented in PureData earlier, it makes sense to mention two big advantages over that:

1. Text-interface, enabling quicker notation of ideas, version-control and a mouseless workflow.
2. Single sample feedback loops, as used in the classicVocoder.

The downsiders of Faust to me are a steep learning curve and error messages that are often very verbose and unclear.

6 Use cases

The author has used VoiceOfFaust mostly for voice transformation in a musical context, but it has also come in handy to turn a bass-guitar into a synth [14].

7 Deployment

VoiceOfFaust heavily leans on knowing the pitch of the input signal. Since it's not yet possible to do decent pitchtracking in Faust, an external pitchtracker which sends the pitch trough OSC is used.

This limits the usable architectures to the ones supporting OSC.

Specifically, it would be nice to have VoiceOfFaust as a plugin within a DAW, but that is not directly possible.

The master slave architecture is a usable workaround.

To compile VoiceOfFaust, run one of the compilation scripts that support OSC, for example:

faust2jack -osc FMvocoder.dsp

To run it, you can use one of the scripts in the launchers directory, for example:

./FMvocoder_PT

This will start puredata with the pitchtracker patch plus a synth, and connect everything trough jack.

8 Acknowledgements

Many thanks to the developers of Faust [1] and Pure Data [2] for making dsp so accesable yet powerfull.

References

[1] http://faust.grame.fr

[2] https://puredata.info

[3] http://www.katjaas.nl/helmholtz/helmholtz.html

[4] http://www.katjaas.nl/compander/compander.html

[5] https://www.nada.kth.se/utbildning/grukth/exjobb/rapportlistor/2010/rapporter10/szabo_adam_10131.pdf

[6] http://forum.pdpatchrepo.info/topic/5992/casio-cz-oscillators

[7] http://msp.ucsd.edu/techniques/v0.11/book-html/node96.html

[8] http://chrischafe.net/glitch-free-fm-vocal-synthesis

[9] http://anasynth.ircam.fr/home/english/media/singing-synthesis-chant-program

[10] https://ccrma.stanford.edu/~mjolsen/220a/fp/Foflet.dsp

[11] https://csound.github.io/docs/manual/fof2.html

[12] https://en.wikipedia.org/wiki/Karplus%E2%80%93Strong_string_synthesis

[13] https://github.com/magnetophon/VoiceOfFaust

[14] http://magnetophon.nl/sounds/BucketBoyz/ShiningBrightLight.mp3