[Personal Notes]

Soesilo Wijono+

Monday, January 19, 2015

Sound Transformation with MTG's Spectral Modeling Synthesis Tools

It is **Spectral Modeling Synthesis sms-tools** of **MTG UPF** (Music Technology Group, Universitat Pompeu Fabra, Barcelona), by Prof. Xavier Serra (also as instructor in the Audio Signal Processing for Music Applications, Coursera).

http://mtg.upf.edu/technologies/sms
https://github.com/MTG/sms-tools
https://github.com/MTG/essentia

Several simple experiments I made.

How to?

1. (Fork and) clone Github's sms-tools into local repository.
2. Install **sms-tools** (and **essentia** if interested) in Ubuntu (or Debian, I don't know whether it works in other Linux distribs, I compiled sms-tools in Windows 7 successfully).
3. For transformation, open `sms-tools/software/transformations` and `sms-tools/software/transformations-interface` folders.
4. Available transformation models: sine, STFT morph (short time Fourier transform), harmonic, stochastic, HPS (harmonic plus stochastic), HPS morph.
5. Either (a) Run `python transformations_GUI.py` or (b) write Python script to call transformation models' functions.
6. In the GUI or Python code, first to do analysis to determine best HPS model's parameters, then apply transformation.
7. To help determine f0, harmonics, and several parameters, we can use "**Sonic Visualiser**" (free) with its Spectogram pane and Aubio plugin (pitch detector, onset detector).

Disclaimer: I'm not musician, just amateur hobbyist in music/sound/audio =)

---

(1) Single-pass.
Source is "speech-female.wav" with duration of 3.2836 sec.
HPS analysis parameters are,

- window type = blackman,
- window size M = 1101,
- FFT size N = 4096,
- threshold t = -90,
- minimum duration of harmonic tracks minSineDur = 0.01,
- minimum f0, minfo = 120,
- maximum f0, maxf0 = 250,
- error threshold, f0et = 8,(to let more error in the TWM algorithm),
- maximum number of harmonics nH = 8, (to minimize maximum number of harmonics for more transformation),
- maximum frequency deviation in harmonic tracks harmdevSlope = 0.2, (big enough to to allow higher harmonics to deviate more than lower harmonics.),
- stochastic decimation/approximation factor stocf = 0.1, (small enough for higher decimation, more compact representation.)

Transformation,
Changing pitches of female to male by scaling the frequency down, reverse the time scaling, to create totally different speech (encrypted / alien sound).

HPS transformation's parameters:
freqScaling
[0, 0.5, 1, 0.5]
freqStretching

[0, 0.99, 1, 0.99]
timbrePreservation
1
timeScaling
[0, 0, 0.1, 1, 0.2, 0.9, 0.3, 0.8, 0.4, 0.7, 0.5, 0.6, 0.6, 0.5, 0.7, 0.4, 0.8, 0.3, 0.9, 0.2, 1, 0.1]
Output WAVE
Alien speech freesound.org

---

(2) Single-pass.
Source is "freesound.org, piano phrase" with duration of 3.8458 sec.
HPS analysis parameters are,

- window type = blackman,
- window size M = 1101,
- FFT size N = 4096,
- threshold t = -80,
- minimum duration of harmonic tracks minSineDur = 0.01,
- minimum f0, minfo = 100,
- maximum f0, maxf0 = 300,
- error threshold, f0et = 5,(to let more error in the TWM algorithm),
- maximum number of harmonics nH = 100,
- maximum frequency deviation in harmonic tracks harmdevSlope = 0.01,
- stochastic decimation/approximation factor stocf = 0.2,

Transformation, Applying square wave to frequency stretching, to produce a swinging transformation. Square wave is sampled with 100 periods at 5 Hz generated by scipy.signal.square, over duration of the sound.

HPS transformation's parameters:
freqScaling
[0, 1, 1, 1]
freqStretching
[0.0000,1.00,0.0385,1.00,0.0769,1.00,0.1154,-1.00,0.1538,-1.00,0.1923,
-1.00,0.2307,1.00,0.2692,1.00,0.3077,-1.00,0.3461,-1.00,0.3846,
-1.00,0.4230,1.00,0.4615,1.00,0.5000,1.00,0.5384,-1.00,0.5769,
-1.00,0.6153,1.00,0.6538,1.00,0.6922,1.00,0.7307,-1.00,0.7692,
-1.00,0.8076,1.00,0.8461,1.00,0.8845,1.00,0.9230,-1.00,0.9615,-1.00,0.9999,
-1.00,1.0384,1.00,1.0768,1.00,1.1153,-1.00,1.1537,-1.00,1.1922,
-1.00,1.2307,1.00,1.2691,1.00,1.3076,-1.00,1.3460,-1.00,1.3845,
-1.00,1.4229,1.00,1.4614,1.00,1.4999,1.00,1.5383,-1.00,1.5768,
-1.00,1.6152,1.00,1.6537,1.00,1.6922,1.00,1.7306,-1.00,1.7691,
-1.00,1.8075,1.00,1.8460,1.00,1.8844,1.00,1.9229,-1.00,1.9614,-1.00,1.9998,
-1.00,2.0383,1.00,2.0767,1.00,2.1152,-1.00,2.1537,-1.00,2.1921,
-1.00,2.2306,1.00,2.2690,1.00,2.3075,-1.00,2.3459,-1.00,2.3844,
-1.00,2.4229,1.00,2.4613,1.00,2.4998,1.00,2.5382,-1.00,2.5767,
-1.00,2.6151,1.00,2.6536,1.00,2.6921,1.00,2.7305,-1.00,2.7690,
-1.00,2.8074,1.00,2.8459,1.00,2.8844,1.00,2.9228,-1.00,2.9613,-1.00,2.9997,
-1.00,3.0382,1.00,3.0766,1.00,3.1151,-1.00,3.1536,-1.00,3.1920,
-1.00,3.2305,1.00,3.2689,1.00,3.3074,-1.00,3.3459,-1.00,3.3843,
-1.00,3.4228,1.00,3.4612,1.00,3.4997,1.00,3.5381,-1.00,3.5766,
-1.00,3.6151,1.00,3.6535,1.00,3.6920,1.00,3.7304,-1.00,3.7689,
-1.00,3.8073,1.00,3.8458,1.00]
timbrePreservation
0
timeScaling[0, 0, 1, 1]
Output WAVE
Swinging piano freesound.org

---

(3) Single-pass.
Source is "freesound.org violin" with duration of 5.0 sec.
HPS analysis parameters are,

- window type = blackman,
- window size M = 661,
- FFT size N = 2048,
- threshold t = -90,
- minimum duration of harmonic tracks minSineDur = 0.01,
- minimum f0, minfo = 400,
- maximum f0, maxf0 = 1000,
- error threshold, f0et = 5,
- maximum number of harmonics nH = 40,

- maximum frequency deviation in harmonic tracks harmdevSlope = 0.01,
- stochastic decimation/approximation factor stocf = 0.1,

Transformation, Applying sawtooth wave to frequency scaling, to produce a mosquito-like sound. Frequency scaling is applied by 40 Hz sawtooth sampled at 100 Hz generated by using scipy.signal.sawtooth, over duration of the sound.

HPS transformation's parameters:
freqScaling
[0.0000,0.50,0.0388,0.62,0.0777,0.52,0.1165,0.65,0.1554,0.55,0.1942,0.67,0.2331,0.57,0.2719,
0.69,0.3108,0.60,0.3496,0.72,0.3885,0.62,0.4273,0.52,0.4662,0.64,0.5050,0.54,0.5439,0.67,
0.5827,0.57,0.6215,0.69,0.6604,0.59,0.6992,0.72,0.7381,0.62,0.7769,0.52,0.8158,0.64,0.8546,
0.54,0.8935,0.66,0.9323,0.57,0.9712,0.69,1.0100,0.59,1.0489,0.71,1.0877,0.61,1.1265,0.51,
1.1654,0.64,1.2042,0.54,1.2431,0.66,1.2819,0.56,1.3208,0.68,1.3596,0.59,1.3985,0.71,1.4373,
0.61,1.4762,0.51,1.5150,0.63,1.5539,0.53,1.5927,0.66,1.6316,0.56,1.6704,0.68,1.7092,0.58,
1.7481,0.71,1.7869,0.61,1.8258,0.51,1.8646,0.63,1.9035,0.53,1.9423,0.65,1.9812,0.55,2.0200,
0.68,2.0589,0.58,2.0977,0.70,2.1366,0.60,2.1754,0.50,2.2143,0.63,2.2531,0.53,2.2919,0.65,
2.3308,0.55,2.3696,0.67,2.4085,0.58,2.4473,0.70,2.4862,0.60,2.5250,0.50,2.5639,0.62,2.6027,
0.52,2.6416,0.65,2.6804,0.55,2.7193,0.67,2.7581,0.57,2.7969,0.70,2.8358,0.60,2.8746,0.72,
2.9135,0.62,2.9523,0.52,2.9912,0.64,3.0300,0.54,3.0689,0.67,3.1077,0.57,3.1466,0.69,3.1854,
0.59,3.2243,0.72,3.2631,0.62,3.3020,0.52,3.3408,0.64,3.3796,0.54,3.4185,0.66,3.4573,0.57,
3.4962,0.69,3.5350,0.59,3.5739,0.71,3.6127,0.61,3.6516,0.51,3.6904,0.64,3.7293,0.54,3.7681,
0.66,3.8070,0.56,3.8458,0.68]
freqStretching
[0, 1, 1, 1]
timbrePreservation
0
timeScaling
[0, 0, 1, 1]
Output WAVE
Violin to mosquito freesound.org

---

(4) Single-pass.
Source is "speech-female.wav" with duration of 3.2836 sec.
HPS analysis parameters are,

- window type = blackman,
- window size M = 1101,
- FFT size N = 4096,
- threshold t = -90,
- minimum duration of harmonic tracks minSineDur = 0.01,
- minimum f0, minfo = 120,
- maximum f0, maxf0 = 250,
- error threshold, f0et = 8,(to let more error in the TWM algorithm),
- maximum number of harmonics nH = 8, (to minimize maximum number of harmonics for more transformation),
- maximum frequency deviation in harmonic tracks harmdevSlope = 0.2, (big enough to to allow higher harmonics to deviate more than lower harmonics.),
- stochastic decimation/approximation factor stocf = 0.1, (small enough for higher decimation, more compact representation.)

Transformation,
To change pitch of every word, up and down intermittently. To change time duration of every word up and down, splitted into words: "this" at 0.72s, "is" at 1.05s, "the" at 1.22s, "v" at 1.85s, "of" at 2.25s, "vendetta"at 3.2836s.

HPS transformation's parameters:
freqScaling
[0, 0.5, 0.72, 2, 1.05, 0.8, 1.22, 1.4, 1.85, 0.6, 2.25, 1.7, 3.2836, 0.4]
freqStretching
[0, 1, 0.5, 1.1, 1, 0.98]
timbrePreservation
1
timeScaling
[0, 0, 0.72, 0.4, 1.05, 1.3, 1.22, 1.4, 1.85, 1.6, 2.25, 1.8, 3.2836, 3.2836]
Output WAVE
Unavailable on the cloud.

---

(5) Multiple sounds, multiple passes, source are all natural sounds,
Sources:
"92004__jcveliz__voilinpart2compress.wav"
https://www.freesound.org/people/jcveliz/sounds/92004/

bass drum = "90150__menegass__bd05.wav"
http://www.freesound.org/people/menegass/sounds/90150/

snare drum 1 = "82238__kevoy__snare-drum.wav"
http://www.freesound.org/people/KEVOY/sounds/82238/

snare drum 2 = "2103__opm__sn-set4.wav"
http://www.freesound.org/people/opm/sounds/2103/

cowbell = "22759__franciscopadilla__56-cowbell.wav"
http://www.freesound.org/people/FranciscoPadilla/sounds/22759/

hi-hat 1 = "67210__akosombo__xbhhopen.wav"
http://www.freesound.org/people/Akosombo/sounds/67210/

hi-hat 2 = "100054__menegass__gui-drum-ch.wav"
http://freesound.org/people/menegass/sounds/100054/

Goal:
To do multiple-passes transformation of a violin sound with certain frequency into a piece of song, by using sms-tools HPS transformation functions. Adding some other transformed percussion sounds to the resulted song.
Phase one: Frequency-scaling is determined by a table of MIDI note numbers in D Dorian scale. Frequency stretching for each beat is randomized. Frequency-scaling of the percussions transformation is randomized for each beat.
Phase two of transformation is time-scaling applied to the composed song, by a table of time periods.

Steps:

- Provide a python-list variable, Notes = a list of MIDI note number, in D Dorian scale.
- Provide a python-list variable, Freqs = convert Notes to its frequency values.
- I do not use transformation GUI, instead call functions directly from, "sms-tools/software/transformations_interface/hpsTransformations_function.py"
- To analyze (HPS), call analysis() function.
- To HPS-transform and synthesis, call transformation_synthesis() function.
- Input file is violin sound with 901.546 Hz of main frequency. Note frequency is tracked by using "Aubio Note Tracker" in Sonic Visualiser.
- Analysis parameters are, window = "blackman" , M = 601 , N = 2048, t = -90 , minSineDur = 0.01 , nH = 100 , minf0 = 400 , maxf0 = 1200 , f0et = 5 , harmDevSlope = 0.01 , stocf = 0.2
- Call "hpsTransformations_function.analysis()" as follow,

```
import hpsTransformations_function as HTF
inputFile, fs, hfreq, hmag, mYst = HTF.analysis(inputFile,
                                window, M, N, t, minSineDur,
                                nH, minf0, maxf0, f0et,
                                harmDevSlope, stocf)
```

Note that, hpsTransformations_function has been edited to remove all plotting commands, also to return output signal "y" in the transformation_synthesis() function.

- Establish a for loop to all Freqs, in each iteration do frequency-scaling transformation according to Freqs[] frequency value.
- Frequency scaling parameter freqScaling is set to [0, scale, 1, scale], in which "scale" is scaling value to mimic the MIDI note frequency.
- In the same time, frequency stretching parameter freqStretching is set with random values, [0, random.uniform(0.8,0.9), 1, random.uniform(0.9,1.0)], to produce different timbre for each beat.
- TimbrePreservation parameter is set to 0, to let timbre changes.
- Do phase-1 transformation and synthesis to the input sound, by calling hpsTransformations_function.py function,

```
# transformation_synthesis() is modified to return y
y = HTF.transformation_synthesis(inputFile, fs, hfreq, hmag, mYst,
                            freqScaling, freqStretching,
                            timbrePreservation, timeScaling)
```

- Append "y" in each iteration to a "song" variable.
- Until this step, variable "song" contains piece of song with melody determined by Freqs' frequencies.
- Next step is to add several percussion instrument sounds to the "song".
- Do HPS-analysis first to each percussion sounds, bass-drum, snare-drum1, snare-drum2, cowbell, hi-hat1, hi-hat2.

- They have low frequencies, single-note and natural, due to the same characteristics, we can use the same analysis parameters to all of them.
- HPS-analysis parameters are, window="blackman", M=801, N=2048, t=-100, minSineDur=0.005, nH=80, minf0=10, maxf0=400, f0et=7, harmDevSlope=0.01, stocf=0.1
- Call "hpsTransformations_function.analysis()" to each percussion sounds, and store the returned values to respective variables.
- Establish a for loop, in each iteration get a "beat" counter for each 8 iterations, so "beat" variable has [0..7] of integer value. This can be done by using python % modulo operator.
- When "beat" has value of 0 and 4, add transformed bass-drum sound. Bass-drum analysis values are used to call "hpsTransformations_function.transformation_synthesis()" function. Frequency scaling parameter is set to a random value, by calling random.uniform() function. So, for each iteration bass-drum has different frequency.
- When "beat" has value of 2 and 6, add transformed snare-drum1 sound. Do the same random transformation as bass-drum.
- For every "beat" (every iteration), add transformed snare-drum2 sound. Do the same as random transformation bass-drum.
- When "beat" has value of 3 and 7, add transformed cowbell sound. Do the same as random transformation bass-drum.
- When "beat" has the same value as random integer [0..7], add transformed hi-hat1 sound. Do the same random transformation as bass-drum.
- When "beat" has value of 5, add transformed hi-hat2 sound. Do the same as random transformation bass-drum.
- Until this step, "song" variable will have a complete piece of the song melody according to the MIDI note numbers specified, with added percussion sounds. Output sound of this phase-1 transformation is saved into a WAVE file, by using "models.utilFunctions.wavwrite()" function.
- In the phase-2 analysis, the output sound of phase-1 is analyzed by using the same parameters as input sound. Call "hpsTransformations_function.analysis()" to the phase-1 output song.
- Do time-scaling transformation, with timeScale parameter is set to a list of values, such that for each beat of the song, the period is scaled by certain factor. Generate timeScale by using a loop over all beats in the "phase-1 song". First pair is of course = 0,0.
- Call "hpsTransformations_function.transformation_synthesis()" to the analysis values.
- Output is the final phase-2 song. Write this song to WAVE file by using "models.utilFunctions.wavwrite()" function.

The final song is "Singing Frogs".

Output sound, (size = 3.876 MB)
Singing frogs freesound.org

Code:
https://github.com/flyingdisc/harmonic-stochastic-sound-transformation

---

Labels: audio signal processing, music, python

Links to this post

Create a Link

Home Older Post

Archive

January 2015 (2)
December 2014 (2)
August 2014 (1)
June 2014 (1)
February 2014 (1)
January 2014 (2)
December 2013 (5)

November 2013 (1)
September 2013 (1)
June 2013 (1)
October 2011 (2)

Simple template. Powered by Blogger.