**Notes on Sound Features toolbox. Version 0.2**

LSS 16 Feb 2014.

Updated 2 April 2014

There’s a lot of .m files in there – probably not all are necessary. Some are not called at all! However, I believe that the functions are complete, in the sense that no other non-standard-MATLAB functions are needed.

The specific calls that matter are:

***setparameters\_monoonsets.m***

example call: setparameters\_monoonsets(sax,'' ,'') ;

(Note that this was preceded by the assignment

sax = '/Users/lss/matlab\_stuff/music/sax' ; )

This is used to set the parameters for creating the AN and Onset files. Parameters for later processing are generally supplied as *varargin* arguments.

Creates the parameter set (in the Matlab file parameters\_monoonset.mat) inside the directory in which the sounds being processed reside (i.e. the string in the variable sax). There’s actually more parameters in there than are used by this software. The ones that matter are

AN.soundlength = 200; % 3 minutes 20 seems to be about the absolute maximum

% 600 is too long, leastwise at 44100 samples, mono

% for experimentation

AN.fmin = 50; % was 150 (was 100): lowest frequency in Gammatone filterbank

AN.fmax = 6500; % was 10000 (was 8000): Highest frequency in Gammatone filterbank

AN.channels = 200; % was 32/64: Number of filterbank channels

AN.N\_erbs = 1 ; % 1 is normal bandwidth. 4 is narrow: bandwidth of channels in filterbank

% parameters for AN signal generation

% AN.minlevel\_zc = 0.0025; % for Cauer filter

AN.minlevel\_zc = 0.0002; % was 0.00014 for Gamma filter: sets sensitivity level for lowest signal: signals less than this are not heard at all.

AN.multiplier = 1.414 ; % root 2: 3dB difference: 2 = 6dB: multiplier for difference between sensitivity levels

AN.iterations = 16 ; %8=24dB 10 is also good: number of sensitivity levels

AN.filtertype = 'gamma'; % filter type: gamma means Gammatone.

% parameters for onsets

% values for wide onset cell

% the way in which the onset works is by a set of depressing synapses feeding into a leaky integrate and fire neuron. The number of such synapses is

2 \* onset.spread\_wide + 1

Appropriate values to use depend on the number of channels in the filterbank, whether one is looking for onsets that are localised in the spectrum , or rather more wideband, ,… For 200 channels, from 50 to 6500 Hz, I find that onset.spread\_wide = 10 is OK. (and in that case onset.onsetcellwt\_wide = 10000/(2\*10 + 1).

onset.onsetcellwt\_wide = 11000/11 ; % weight to onset cell was 1500 for spread-wide = 2 (was 10000 for single AN input): this sets the weight between the AN spike and the onset cell. The more AN spikes that innervate the onset cell, the lower this should be.

onset.spread\_wide = 5 ; % number of AN fibers on each side of centre (0->no spread) (was 2)

%

% Note that the above parameters interact: as spread\_wide is increased, so

% the number of synapses on to each onset cell increases, with the total

% number being 2\*onset.spread\_wide + 1. The weight used needs to reflect

% this, because all the weights are (currently) the same, being set to

% onset.onsetcellwt\_wide.

% the next parameter is about exactly how the depressing synapses work. It sets up how the reservoirs on the depressing synapse interact. Best left as is.

onset.depsynparams\_wide = [100 1100 9]; % (was 100 1100 9 )depressing synapse parameters

% the parameters below set up the dissipation level for the leaky integrate-and-fire neurons. Probably best left as is.

onset.dissconst\_wide =0.15; % dissipation

onset.fmaxdiss\_wide = 3500 ; % was 1500: relates to change in dissipation at high frequencies

onset.fmindiss\_wide = 500 ; : relates to change in dissipation at low frequencies.

onset.th\_iandf\_wide = 1 ; % threshold

onset.rp\_wide = 0.02 ; % was 0.015 (0.003 is an attempt at AM): refractory perions

onset.rrp\_wide = 0.015 ; % not used

% note that ITD.mincluster\_b etc. is used in ITD calculations for wide

% onset cells

% below: used in overallonset (which generates onset intervals – the periods of time during which an onset is taking place)

onset.sensitivitygap = 0.005 ; % max gap between sensitivity levels

onset.intervalgap = 0.015 ; % used in overallonset for separating onsets

***runTIMIT250gabor3\_fn***

This is the call that calls all the other functions, and is the subject of a separate note. A typical call might look like

runTIMIT250gabor3\_fn(timitlocn, 'filelist\_all.txt', 'ANSigdir', 'AMMeanPlain250b', 'onsetSigdir', 'onsetMeans250b', gaborparamsall, 'Vectors250c', 'gabordirs', gabordirs, 'runan', 0, 'runonsets', 0, 'runplainandonsets', 0, 'rungabors', 0, 'filelist\_male','filelist\_male.txt', 'filelist\_female', 'filelist\_female.txt' ,'segmentsummary', 0 , 'runPCAICA', 1, 'rungeneratesegments', 0, 'n\_icas', 200, 'n\_pcas', 40, 'runclustering\_pca', 1) ;

It calls a large number of other functions, some detailed below.

***create\_AN\_files.m***

example call: n1 = create\_AN\_files(sax, '','', 'filelist.txt', storebmm , 'bmSigdir','ANSigdir') ;

Creates the AN files from a set of sounds, where the sounds are in the files named in the file [sax ‘/filelist.txt’]. The file names should be one per line: they may be .wav or .au files, using the parameters in parameters\_monoonset.mat. The sound files to be processed are found in the directory in the string in the variable sax. If storebmm is 0, the gammatone filterbank is called, but the files are not saved. If it is 1, the gammatone filterbank is called, and the bmm files (the relevant output from the filterbank) are saved in the directory [sax ‘/bmSigdir’]. If storebmm is 2, the gammatone filterbank is not called, and the bmm files are loaded from the directory [sax ‘/bmSigdir’] '. In any case, the output from the function, namely the AN-like spike trains, are placed in the directory [sax ‘/bmSigdir’].

***generateonsetspikes2\_mono***

Example call: generateonsetspikes2\_mono(sax, '','', 'filelist.txt', 'ANSigdir', 'onsetSigdir', display) ;

Creates the Onset spikes from the AN files, using the parameters in parameters\_monoonset.mat. The files to be used are those named in filelist.txt, in the directory [sax ‘/ANSigdir’]. (Note that the file stem is used – so the same list of filenames can be used for both create\_AN\_files.m and generateonsetspikes2\_mono.m: there is one restriction, and that is the file stem can only contain one ‘.’ character). The onset spikes produced are stored in the directory [sax ‘/onsetSigdir’].

Note that this function can be slow. It involves a great deal of calculation for the values at the depressing synapses of the onset units. (The total number of these will be approximately (2 \* onset.spread\_wide + 1) \* AN.channels \* AN.iterations: and the value in each one will be calculated at each sample time.)

*A reasonably detailed description of the processing for the whole of the above may be found in Newton’ and Smith’s JASA paper from 2012, at* [*http://www.cs.stir.ac.uk/~lss/recentpapers/JASA2012NewtonSmith.pdf*](http://www.cs.stir.ac.uk/~lss/recentpapers/JASA2012NewtonSmith.pdf) .

***generatemeanplain\_mono***

Example call: generatemeanplain\_mono(sax, '','', 'filelist.txt', 'ANSigDir', 'meanplain', resamplerate, display) ;

Creates the “level 0” representation from the AN-like spikes in the files named in [sax ‘/filelist.txt’] (again, the file stem is used), to be found in the directory [sax ‘/ANSigDir’], and places the outputs in the directory [sax ‘/'meanplain']. The signals are downsampled to resamplerate. What is stored are .mat arrays called 'mlm', 'ANParams', 'mlmparams': mlm contains the actual output as an array which is a number of samples (at resamplerate) by a number of channels (the number used originally when the AN files were created). ANParams contains the parameters used in creating the AN file, and mlmparams contains the parameters used in this call. The parameter display may have the value 0 (no display), or 1, in which case a display of the mlm array is produced. No varargin parameters are currently used, although the skeleton code for these is included.

***generatemeanlevelonset\_mono***

Example call:generatemeanlevelonset\_mono(sax, '','', 'filelist.txt', 'onsetSigDir', 'onsetoutputs', resamplerate, display);

Creates the “level 1” representation from the onset-like spikes in the files named in [sax ‘/filelist.txt’] (again, the file stem is used), to be found in the directory [sax ‘/onsetSigDir’], and places the outputs in the directory [sax ‘/'onsetSigDir']. The signals are downsampled to resamplerate.

What is actually stored is 'mlo', 'onsetparams', 'ANParams', 'mloparams': mlo is the array (again, number of samples (at resamplerate) by a number of channels (the number used originally when the AN files were created)); onsetparams and ANParams are the parameters used when the AN files, and the onset files were created, and mloparams are the parameters in this call. The parameter display may have the value 0 (no display), or 1, in which case a display of the mlo array is produced. No varargin parameters are currently used, although the skeleton code for these is included.

***generategaboroutputs\_mono***

Example call: generategaboroutputs\_mono(sax, '','', 'filelist.txt', 'ANSigDir', 'gaborOutput', gaborparams, resamplerate, display)

Creates the “level 1” representation of the sound from the AN-like spikes in the files named in [sax ‘/filelist.txt’] (again, the file stem is used), to be found in the directory [sax ‘/ANSigDir’], and places the outputs in the directory [sax ‘/gaborOutput’]. The signals are down-sampled to resamplerate.

The function calls runmultigabors.m, with certain parameters (specifically type, which is set to 3) fixed. This in turn calls gabor\_fn\_1.m to create the Gabor windows, and these are then convolved, channel by channel with a continuous signal recreated from the spike trains. The actual Gabor functions used can be displayed by setting display to a value greater than 1. As the program stands, the Gabor functions are displayed for each file (even although they will all be the same).

Were the function to have implemented Gabor functions precisely, the output would be both positive and negative (setting type in the above paragraph to a value <= 2 would have this effect). By setting type to 3 or 4, the Gabor convolutions calculated and then added as absolute values, resulting in the overall output being nonnegative. Note that the Gabor functions are computed at a sampling rate of resample\_gabor, which defaults to 2000 samples/second (or one sample per 0.5ms).

The parameter gaborparams is a structure containing the values that define the Gabor function. These are bandwith, lambda, gamma and theta: for a look at how they are used, see the function that creates the Gabor filter, namely gabor\_fn\_1.m In essence, bandwidth sets the size of the Gaussian, lambda sets the frequency of the sin wave inside the Gaussian (in units of 1/resample\_gabor, which defaults to 0.5ms), gamma sets the aspect ratio (i.e. the ratio of the size of the Gaussian axes in the *x* (time) and *y* (frequency band) direction), theta sets the angle of the sine wave. To make the sin wave parallel to the time (*x*) axis (for example to detect amplitude modulation), set theta to π/2. (Note that the phase value is set to 0, so that the Gabor function has a peak at the centre of the Gaussian). With a signal with 200 channels, a value for bandwidth of 0.7, for gamma of 4 seem appropriate for AM detection. The value of lambda to use depends on the frequency of the AM expected: for 200 Hz, at a resample\_gabor rate of 2000, a value of 10 would be appropriate (10 = (2000/200): i.e. the period of the signal of interest in 1/2000 second units).

What is actually stored is 'gaboroutput', 'ANParams', and 'gaborparams': 'gaboroutput'is the array (again, number of samples (at resamplerate) by a number of channels (the number used originally when the AN files were created)); ANParams are the parameters used when the AN files were created, and gaborparams are the parameters in this call (both those set in the command line, and others used). The parameter display may have the value 0 (no display), or 1, in which case a display of the Gabor output array is produced, or 2, in which case the Gabor functions used are also displayed.

This function has a number of *varargin* parameters:

'resample\_gabor': this may be used to alter the internal resampling rate at which the Gabor functions are calculated. Defaults to 2000.

'zeromean': this may be used to ensure that the sun of the values inside the Gabor filter is 0. Normally, this will be (more or less) the case, but by using ‘zeromean’, 1 in the call, this can be enforced. Defaults to 0.

'szxvalue': used to set the number of bandpass channels that are in the Gabor filter. The best value to use will depend on the number of channels used in the gammatone filterbank, as well as on the application in hand. Defaults to 13, which seems a reasonable value when there are 100 to 200 filters covering the audio spectrum.

Thus a different example call might be

generategaboroutputs\_mono(sax, '','', 'filelist.txt', 'ANSigDir', 'gaborOutput', gaborparams, 100, 2, 'szxvalue', 7, 'zeromean', 1) ;

*Notes*:

1. This document describes only the top-level functions. It should be extended to say more about the lower level ones.
2. The easiest way to use the system is to run the spike generating functions first, for all the sounds, and then to use the other functions on those. The spike generating functions can be quite slow (particularly the onset spike generator).
3. Some of the functions are from elsewhere. Specifically, erb.m, bwcorrection.m, ErbRateToHz.m, HzToErbRate.m, MakeErbCFs.m, gammatone1.m, originate from the University of Sheffield’s MAD toolbox, and spikeraster.m was written by David Sterratt, now of the University of Edinburgh. The function AN\_coder\_GRM.m was written by Gabriel Reines, who was studying at the University of Edinburgh, based on an earlier function by Leslie Smith.