The Open Master Hearing Aid (openMHA)

4.5.0

Documentation of openMHA plugins (openMHA)

The Open Master Hearing Aid (openMHA) – Documentation of openMHA plugins (open-MHA)

HörTech gGmbH
Marie-Curie-Str. 2
D–26129 Oldenburg

Contents

1 Plugin category 'AC-variables'

1.1 acmon

This algorithm converts AC variables into parsable monitor variables.

1.1.0.1 Supported domains

The MHA plugin `acmon` supportes these signal domains:

- waveform to waveform
- spectrum to spectrum

1.1.0.2 Categories

AC-variables

1.1.0.3 Configuration variables

Name	Type	Description	Default
mhaconfig_in	parser	Input configuration	(see below)
mhaconfig_out	parser	Output configuration	(see below)
varlist	vector<string>	complete list of variables	(monitor)
dimensions	vector<string>	variable dimensions in AC space	(monitor)
dispmode	keyword_list	display mode of variables **Range:** [vector matrix]	vector
recmode	keyword_list	record mode **Range:** [cont snapshot]	cont

Variables of sub-parser `mhaconfig_in`:

Name	Type	Description	Default
channels	int	Number of audio channels	(monitor)
domain	string	Signal domain (MHA_WAVEFORM or MHA_SPECTRUM)	(monitor)
fragsize	int	Fragment size of waveform data	(monitor)
wndlen	int	Window length of spectral data	(monitor)
fftlen	int	FFT length of spectral data	(monitor)
srate	float	Sampling rate in Hz	(monitor)

Variables of sub-parser `mhaconfig_out`:

Name	Type	Description	Default
channels	int	Number of audio channels	(monitor)
domain	string	Signal domain (MHA_WAVEFORM or MHA_SPECTRUM)	(monitor)
fragsize	int	Fragment size of waveform data	(monitor)
wndlen	int	Window length of spectral data	(monitor)
fftlen	int	FFT length of spectral data	(monitor)
srate	float	Sampling rate in Hz	(monitor)

1.2 acsave

Save chain data to text or Matlab 4 files.

Usage:

1. set up file name and type

2. set the maximal length. This will start the recording.

3. Set "flush" to yes to save recorded frames. This will overwrite previously written data.

File name and type can be changed at any time and has to be valid when sending the flush command. Changing the list of variables also starts the recording with the currently configured recording length (previously recorded data might be overwritten). Issueing the 'flush' command frees allocated memory.

1.2.0.1 Detailed description

The 'acsave' plugin can save numeric algorithm communication variables (AC variables) into files. The files can have plain text, MATLAB 4.x or MATLAB script format. Each signal frame represents a row. The number of columns is gathered at preparation time. If a variable size is increased after preparation, only the part available at preparation time is stored. If the size is decreased, it is zero-padded to the original size.

To save the data to disk, first set up file name and type. Then setting the maximal length will start the recording. At any time, set 'flush' to yes in order to save the recorded frames. This will overwrite previously written data.

File name and type can be changed at any time and have to be valid when sending the flush command.

1.2.0.2 Supported domains

The MHA plugin `acsave` supportes these signal domains:

- waveform to waveform
- spectrum to spectrum

1.2.0.3 Categories

AC-variables

1.2.0.4 Configuration variables

Name	Type	Description	Default
mhaconfig_in	parser	Input configuration	(see below)
mhaconfig_out	parser	Output configuration	(see below)
fileformat	keyword_list	file format of output file **Range:** [txt mat4 m]	txt
name	string	output file name	
reclen	float	maximal recording length in seconds **Range:** [0,]	10
flush	bool	flush the buffers to disk	no
vars	vector<string>	list of variables to be saved (empty: save all)	[]

Variables of sub-parser `mhaconfig_in`:

Name	Type	Description	Default
channels	int	Number of audio channels	(monitor)
domain	string	Signal domain (MHA_WAVEFORM or MHA_SPECTRUM)	(monitor)
fragsize	int	Fragment size of waveform data	(monitor)
wndlen	int	Window length of spectral data	(monitor)
fftlen	int	FFT length of spectral data	(monitor)
srate	float	Sampling rate in Hz	(monitor)

Variables of sub-parser `mhaconfig_out`:

Name	Type	Description	Default
channels	int	Number of audio channels	(monitor)
domain	string	Signal domain (MHA_WAVEFORM or MHA_SPECTRUM)	(monitor)
fragsize	int	Fragment size of waveform data	(monitor)
wndlen	int	Window length of spectral data	(monitor)
fftlen	int	FFT length of spectral data	(monitor)
srate	float	Sampling rate in Hz	(monitor)

1.3 acSteer

Steering Vector Loading Plugin

1.3.0.1 Detailed description

The `acSteer` plugin loads a file contaning pre-computed steering filters (e.g. MVDR filters) to be used within a beamformer. The steering filters can be monaural (**nrefmic = 1**) or binaural (**nrefmic = 2**). The whole file consists of a column vector of concatenated steering vectors, which are formatted in the order of **angle** and **channel**. This means that the first channel vector of the first angle is followed by the second channel vector of the first angle until the last channel. The channel vectors of the first angle are followed by the channel vectors of the second angle and so an and so forth.

If the steering filters have been computed for two reference microphones, the steering filters of the second reference microphone just follow the ones for the first microphone and have the same format.

This plugin is typically located between a localization plugin (e.g. `doasvm_classification`) and a beamforming plugin (e.g. `steerbf`). The localization plugin estimates the source direction and saves it in an AC variable. This plugin reads the saved direction from the corresponding AC variable and saves the corresponding steering vector to the AC space, which is used by the succeeding beamforming plugin for steering the beam towards that particular direction.

The configuration variable **nrefmic** indicates the number of different reference microphone settings, for which the filters were computed. For each reference microphone and each possible DOA angle and each input channel one filter should be provided so that

$$nsteerchan = nrefmix * nchan * nangle \tag{1}$$

1.3.0.2 Supported domains

The MHA plugin `acSteer` supportes these signal domains:

- spectrum to spectrum

1.3.0.3 Categories

AC-variables

1.3.0.4 Configuration variables

Name	Type	Description	Default
mhaconfig_in	parser	Input configuration	(see below)
mhaconfig_out	parser	Output configuration	(see below)
steerFile	string	Name of the input file where the steering vectors are saved	steerfile.bin
acSteerName1	string	Name of the AC variable where the steering vectors of the first (left) reference microphone are saved	acSteerLeft
acSteerName2	string	Name of the AC variable where the steering vectors of the second (right) reference microphone are saved	acSteerRight
nsteerchan	int	Number of channels in each steering vector	4
nrefmic	int	Number of reference microphones **Range:**]0,2]	1

Variables of sub-parser `mhaconfig_in`:

Name	Type	Description	Default
channels	int	Number of audio channels	(monitor)
domain	string	Signal domain (MHA_WAVEFORM or MHA_SPECTRUM)	(monitor)
fragsize	int	Fragment size of waveform data	(monitor)
wndlen	int	Window length of spectral data	(monitor)
fftlen	int	FFT length of spectral data	(monitor)
srate	float	Sampling rate in Hz	(monitor)

Variables of sub-parser `mhaconfig_out`:

Name	Type	Description	Default
channels	int	Number of audio channels	(monitor)
domain	string	Signal domain (MHA_WAVEFORM or MHA_SPECTRUM)	(monitor)
fragsize	int	Fragment size of waveform data	(monitor)
wndlen	int	Window length of spectral data	(monitor)
fftlen	int	FFT length of spectral data	(monitor)
srate	float	Sampling rate in Hz	(monitor)

2 Plugin category 'beamforming'

2.1 adm

Adaptive differential microphone

2.1.0.1 Supported domains

The MHA plugin `adm` supportes these signal domains:

- waveform to waveform

2.1.0.2 Categories

beamforming multichannel

2.1.0.3 Configuration variables

Name	Type	Description	Default
mhaconfig_in	parser	Input configuration	(see below)
mhaconfig_out	parser	Output configuration	(see below)
front_channels	vector<int>	Channel indices for front microphones **Range:** [0,[	[0 1]
rear_channels	vector<int>	Channel indices for rear microphones **Range:** [0,[	[2 3]
distances	vector<float>	distance between front and rear microphones **Range:** [0.0008,0.08]	[0.0108 0.0108]
lp_order	int	Filter order of FIR lowpass filter **Range:** [46,128]	46
decomb_order	int	Filter order of FIR comb compensation filter **Range:** [46,128]	54
bypass	int	if 1, output front microphones directly, if 2, output rear microphones **Range:** [0,2]	0
beta	float	Explicit fixed beta (-1 for adaptive filtering)	-1
coeff_lp	vector<float>	lp coefficients	(monitor)
coeff_decomb	vector<float>	decomb coefficients	(monitor)

Variables of sub-parser `mhaconfig_in`:

Name	Type	Description	Default
channels	int	Number of audio channels	(monitor)
domain	string	Signal domain (MHA_WAVEFORM or MHA_SPECTRUM)	(monitor)
fragsize	int	Fragment size of waveform data	(monitor)
wndlen	int	Window length of spectral data	(monitor)
fftlen	int	FFT length of spectral data	(monitor)
srate	float	Sampling rate in Hz	(monitor)

Variables of sub-parser `mhaconfig_out`:

Name	Type	Description	Default
channels	int	Number of audio channels	(monitor)
domain	string	Signal domain (MHA_WAVEFORM or MHA_SPECTRUM)	(monitor)
fragsize	int	Fragment size of waveform data	(monitor)
wndlen	int	Window length of spectral data	(monitor)
fftlen	int	FFT length of spectral data	(monitor)
srate	float	Sampling rate in Hz	(monitor)

2.2 steerbf

Steerable Beamformer

2.2.0.1 Detailed description

Implements frequency-domain beamformer processing (filter and sum) using externally provided filters. A plugin called `acSteer` can be used to provide the filter coefficients. The filter coefficients to be read are saved as a waveform object in the AC space. Each channel of this object corresponds to a different steering angle. The steering angle is typically determined in real-time by a localization plugin (e.g. `doasvm_classification`). In this case, the index to the corresponding steering direction is read from the AC space. Note that the number of available filters should be consistent with the number of possible steering directions to be estimated. The configuration variable **angle_src** keeps the name of the AC variable for the estimated steering direction. The steering angle can also be fixed in the configuration time using the configuration variable **angle_ind**.

2.2.0.2 Supported domains

The MHA plugin `steerbf` supportes these signal domains:

- spectrum to spectrum

2.2.0.3 Categories

beamforming

2.2.0.4 Configuration variables

Name	Type	Description	Default
mhaconfig_in	parser	Input configuration	(see below)
mhaconfig_out	parser	Output configuration	(see below)
bf_src	string	Provides the beamforming filters encoded as a block matrix: [chanXnangle,nfreq].	
angle_ind	int	Sets the steering angle in filtering. **Range:** [0,1000]	0
angle_src	string	If initialized, provides an int-AC variable of steering index.	

Variables of sub-parser `mhaconfig_in`:

Name	Type	Description	Default
channels	int	Number of audio channels	(monitor)
domain	string	Signal domain (MHA_WAVEFORM or MHA_SPECTRUM)	(monitor)
fragsize	int	Fragment size of waveform data	(monitor)
wndlen	int	Window length of spectral data	(monitor)
fftlen	int	FFT length of spectral data	(monitor)
srate	float	Sampling rate in Hz	(monitor)

Variables of sub-parser `mhaconfig_out`:

Name	Type	Description	Default
channels	int	Number of audio channels	(monitor)
domain	string	Signal domain (MHA_WAVEFORM or MHA_SPECTRUM)	(monitor)
fragsize	int	Fragment size of waveform data	(monitor)
wndlen	int	Window length of spectral data	(monitor)
fftlen	int	FFT length of spectral data	(monitor)
srate	float	Sampling rate in Hz	(monitor)

3 Plugin category 'binaural'

3.1 coherence

Coherence filter

3.1.0.1 Supported domains

The MHA plugin `coherence` supportes these signal domains:

- spectrum to spectrum

3.1.0.2 Categories

binaural filter noise-reduction

3.1.0.3 Configuration variables

Name	Type	Description	Default
mhaconfig_in	parser	Input configuration	(see below)
mhaconfig_out	parser	Output configuration	(see below)
unit	keyword_list	Frequency unit **Range:** [Hz kHz Oct Oct/3 Bark Erb ERB_Glasberg1990]	Hz
f	vector<float>	Frequencies	[]
f_hz	vector<float>	Frequencies in Hz	(monitor)
fscale	keyword_list	frequency scale of filter bank **Range:** [linear bark log erb ERB_Glasberg1990]	linear
ovltype	keyword_list	filter overlap type **Range:** [rect linear hanning exp gauss]	rect
plateau	float	relative plateau width **Range:** [0,1[	0
ftype	keyword_list	frequency entry type **Range:** [center edge]	center
normalize	bool	normalize broadband output amplitude	no
fail_on_nonmonotonic	bool	Fail if frequency entries are non-monotonic (otherwise sort)	yes
fail_on_unique_bins	bool	Fail if center frequencies share the same FFT bin.	yes
cf	vector<float>	final center frequencies in Hz	(monitor)
ef	vector<float>	final edge frequencies in Hz	(monitor)
cLTASS	vector<float>	Bandwidth level correction for LTASS noise in dB	(monitor)
shapes	matrix<float>	Frequency band shapes	(monitor)
tau_unit	keyword_list	tau unit **Range:** [seconds periods]	seconds
tau	vector<float>	Averaging time constant **Range:** [0,]	[0.04]
alpha	vector<float>	Gain exponent **Range:** [0,]	[1]
limit	float	gain limit / dB (zero: no limit) **Range:** [,0]	0
mapping	vector<float>	mapping interval of coherence estimator to coherence (min max) **Range:** [0,1]	[0 1]
average	keyword_list	average mode **Range:** [ipd spec]	ipd
invert	bool	Invert filter after mapping, before exponent.	no
ltgcomp	bool	Long term gain compensation?	no
ltgtau	vector<float>	Long term gain estimation time constant / s **Range:** [0,]	[1]
staticgain	vector<float>	Static gain in frequency bands / dB	[0]
delay	int	Delay between analysis and filter (delay of gains), in fragments. **Range:** [0,]	0

Variables of sub-parser `mhaconfig_in`:

Name	Type	Description	Default
channels	int	Number of audio channels	(monitor)
domain	string	Signal domain (MHA_WAVEFORM or MHA_SPECTRUM)	(monitor)
fragsize	int	Fragment size of waveform data	(monitor)
wndlen	int	Window length of spectral data	(monitor)
fftlen	int	FFT length of spectral data	(monitor)
srate	float	Sampling rate in Hz	(monitor)

Variables of sub-parser `mhaconfig_out`:

Name	Type	Description	Default
channels	int	Number of audio channels	(monitor)
domain	string	Signal domain (MHA_WAVEFORM or MHA_SPECTRUM)	(monitor)
fragsize	int	Fragment size of waveform data	(monitor)
wndlen	int	Window length of spectral data	(monitor)
fftlen	int	FFT length of spectral data	(monitor)
srate	float	Sampling rate in Hz	(monitor)

4 Plugin category 'delay'

4.1 delaysum

delay and sum plugin

4.1.0.1 Supported domains

The MHA plugin `delaysum` supportes these signal domains:

- waveform to waveform

4.1.0.2 Categories

delay and sum

4.1.0.3 Configuration variables

Name	Type	Description	Default
mhaconfig_in	parser	Input configuration	(see below)
mhaconfig_out	parser	Output configuration	(see below)
weights	vector<float>	weights of channels **Range:** [0,[	[1 1]
delay	vector<int>	delay in number of frames **Range:** [0,[	[0 0]

Variables of sub-parser `mhaconfig_in`:

Name	Type	Description	Default
channels	int	Number of audio channels	(monitor)
domain	string	Signal domain (MHA_WAVEFORM or MHA_SPECTRUM)	(monitor)
fragsize	int	Fragment size of waveform data	(monitor)
wndlen	int	Window length of spectral data	(monitor)
fftlen	int	FFT length of spectral data	(monitor)
srate	float	Sampling rate in Hz	(monitor)

Variables of sub-parser `mhaconfig_out`:

Name	Type	Description	Default
channels	int	Number of audio channels	(monitor)
domain	string	Signal domain (MHA_WAVEFORM or MHA_SPECTRUM)	(monitor)
fragsize	int	Fragment size of waveform data	(monitor)
wndlen	int	Window length of spectral data	(monitor)
fftlen	int	FFT length of spectral data	(monitor)
srate	float	Sampling rate in Hz	(monitor)

5 Plugin category 'example'

5.1 example1

5.1.0.1 Detailed description

The **simplest** example of an openMHA plugin.

This plugin scales one channel of the input signal, working in the time domain.

5.1.0.2 Supported domains

The MHA plugin `example1` supportes these signal domains:

- waveform to waveform

5.1.0.3 Categories

example

5.1.0.4 Configuration variables

Name	Type	Description	Default
mhaconfig_in	parser	Input configuration	(see below)
mhaconfig_out	parser	Output configuration	(see below)

Variables of sub-parser `mhaconfig_in`:

Name	Type	Description	Default
channels	int	Number of audio channels	(monitor)
domain	string	Signal domain (MHA_WAVEFORM or MHA_SPECTRUM)	(monitor)
fragsize	int	Fragment size of waveform data	(monitor)
wndlen	int	Window length of spectral data	(monitor)
fftlen	int	FFT length of spectral data	(monitor)
srate	float	Sampling rate in Hz	(monitor)

Variables of sub-parser `mhaconfig_out`:

Name	Type	Description	Default
channels	int	Number of audio channels	(monitor)
domain	string	Signal domain (MHA_WAVEFORM or MHA_SPECTRUM)	(monitor)
fragsize	int	Fragment size of waveform data	(monitor)
wndlen	int	Window length of spectral data	(monitor)
fftlen	int	FFT length of spectral data	(monitor)
srate	float	Sampling rate in Hz	(monitor)

5.2 example2

This plugin multiplies the sound signal in one audio channel by a factor

5.2.0.1 Supported domains

The MHA plugin `example2` supportes these signal domains:

- waveform to waveform

5.2.0.2 Categories

example

5.2.0.3 Configuration variables

Name	Type	Description	Default
mhaconfig_in	parser	Input configuration	(see below)
mhaconfig_out	parser	Output configuration	(see below)
channel	int	Index of audio channel to scale. Indices start from 0. **Range:** [0,[	0
factor	float	The scaling factor that is applied to the selected channel. **Range:** [0,[	0.1

Variables of sub-parser `mhaconfig_in`:

Name	Type	Description	Default
channels	int	Number of audio channels	(monitor)
domain	string	Signal domain (MHA_WAVEFORM or MHA_SPECTRUM)	(monitor)
fragsize	int	Fragment size of waveform data	(monitor)
wndlen	int	Window length of spectral data	(monitor)
fftlen	int	FFT length of spectral data	(monitor)
srate	float	Sampling rate in Hz	(monitor)

Variables of sub-parser `mhaconfig_out`:

Name	Type	Description	Default
channels	int	Number of audio channels	(monitor)
domain	string	Signal domain (MHA_WAVEFORM or MHA_SPECTRUM)	(monitor)
fragsize	int	Fragment size of waveform data	(monitor)
wndlen	int	Window length of spectral data	(monitor)
fftlen	int	FFT length of spectral data	(monitor)
srate	float	Sampling rate in Hz	(monitor)

5.3 example3

This plugin multiplies the sound signal in one audio channel by a factor

5.3.0.1 Supported domains

The MHA plugin `example3` supportes these signal domains:

- waveform to waveform

5.3.0.2 Categories

example

5.3.0.3 Configuration variables

Name	Type	Description	Default
mhaconfig_in	parser	Input configuration	(see below)
mhaconfig_out	parser	Output configuration	(see below)
channel	int	Index of audio channel to scale. Indices start from 0. Only channels with even indices may be scaled. **Range:** [0,[	0
factor	float	The scaling factor that is applied to the selected channel. **Range:** [0,[	0.1
prepared	int	State of this plugin: 0 = unprepared, 1 = prepared	(monitor)

Variables of sub-parser `mhaconfig_in`:

Name	Type	Description	Default
channels	int	Number of audio channels	(monitor)
domain	string	Signal domain (MHA_WAVEFORM or MHA_SPECTRUM)	(monitor)
fragsize	int	Fragment size of waveform data	(monitor)
wndlen	int	Window length of spectral data	(monitor)
fftlen	int	FFT length of spectral data	(monitor)
srate	float	Sampling rate in Hz	(monitor)

Variables of sub-parser `mhaconfig_out`:

Name	Type	Description	Default
channels	int	Number of audio channels	(monitor)
domain	string	Signal domain (MHA_WAVEFORM or MHA_SPECTRUM)	(monitor)
fragsize	int	Fragment size of waveform data	(monitor)
wndlen	int	Window length of spectral data	(monitor)
fftlen	int	FFT length of spectral data	(monitor)
srate	float	Sampling rate in Hz	(monitor)

5.4 example4

This plugin multiplies the sound signal in one audio channel by a factor. It works in the spectral domain.

5.4.0.1 Supported domains

The MHA plugin `example4` supportes these signal domains:

- spectrum to spectrum

5.4.0.2 Categories

example

5.4.0.3 Configuration variables

Name	Type	Description	Default
mhaconfig_in	parser	Input configuration	(see below)
mhaconfig_out	parser	Output configuration	(see below)
channel	int	Index of audio channel to scale. Indices start from 0. Only channels with even indices may be scaled. **Range:** [0,[	0
factor	float	The scaling factor that is applied to the selected channel. **Range:** [0,[	0.1
prepared	int	State of this plugin: 0 = unprepared, 1 = prepared	(monitor)

Variables of sub-parser `mhaconfig_in`:

Name	Type	Description	Default
channels	int	Number of audio channels	(monitor)
domain	string	Signal domain (MHA_WAVEFORM or MHA_SPECTRUM)	(monitor)
fragsize	int	Fragment size of waveform data	(monitor)
wndlen	int	Window length of spectral data	(monitor)
fftlen	int	FFT length of spectral data	(monitor)
srate	float	Sampling rate in Hz	(monitor)

Variables of sub-parser `mhaconfig_out`:

Name	Type	Description	Default
channels	int	Number of audio channels	(monitor)
domain	string	Signal domain (MHA_WAVEFORM or MHA_SPECTRUM)	(monitor)
fragsize	int	Fragment size of waveform data	(monitor)
wndlen	int	Window length of spectral data	(monitor)
fftlen	int	FFT length of spectral data	(monitor)
srate	float	Sampling rate in Hz	(monitor)

5.5 example5

example plugin configuration structure

5.5.0.1 Supported domains

The MHA plugin `example5` supportes these signal domains:

- spectrum to spectrum

5.5.0.2 Categories

example

5.5.0.3 Configuration variables

Name	Type	Description	Default
mhaconfig_in	parser	Input configuration	(see below)
mhaconfig_out	parser	Output configuration	(see below)
channel	int	channel number to be scaled **Range:** [0,[	0
factor	float	scale factor **Range:** [0,2]	1

Variables of sub-parser `mhaconfig_in`:

Name	Type	Description	Default
channels	int	Number of audio channels	(monitor)
domain	string	Signal domain (MHA_WAVEFORM or MHA_SPECTRUM)	(monitor)
fragsize	int	Fragment size of waveform data	(monitor)
wndlen	int	Window length of spectral data	(monitor)
fftlen	int	FFT length of spectral data	(monitor)
srate	float	Sampling rate in Hz	(monitor)

Variables of sub-parser `mhaconfig_out`:

Name	Type	Description	Default
channels	int	Number of audio channels	(monitor)
domain	string	Signal domain (MHA_WAVEFORM or MHA_SPECTRUM)	(monitor)
fragsize	int	Fragment size of waveform data	(monitor)
wndlen	int	Window length of spectral data	(monitor)
fftlen	int	FFT length of spectral data	(monitor)
srate	float	Sampling rate in Hz	(monitor)

5.6 example6

example plugin configuration structure

5.6.0.1 Supported domains

The MHA plugin `example6` supportes these signal domains:

- waveform to waveform

5.6.0.2 Categories

example

5.6.0.3 Configuration variables

Name	Type	Description	Default
mhaconfig_in	parser	Input configuration	(see below)
mhaconfig_out	parser	Output configuration	(see below)
channel	int	channel in which the RMS level is measured **Range:** [0,[	0

Variables of sub-parser `mhaconfig_in`:

Name	Type	Description	Default
channels	int	Number of audio channels	(monitor)
domain	string	Signal domain (MHA_WAVEFORM or MHA_SPECTRUM)	(monitor)
fragsize	int	Fragment size of waveform data	(monitor)
wndlen	int	Window length of spectral data	(monitor)
fftlen	int	FFT length of spectral data	(monitor)
srate	float	Sampling rate in Hz	(monitor)

Variables of sub-parser `mhaconfig_out`:

Name	Type	Description	Default
channels	int	Number of audio channels	(monitor)
domain	string	Signal domain (MHA_WAVEFORM or MHA_SPECTRUM)	(monitor)
fragsize	int	Fragment size of waveform data	(monitor)
wndlen	int	Window length of spectral data	(monitor)
fftlen	int	FFT length of spectral data	(monitor)
srate	float	Sampling rate in Hz	(monitor)

6 Plugin category ’feedback’

6.1 fshift_hilbert

Pitch shifter

6.1.0.1 Supported domains

The MHA plugin `fshift_hilbert` supportes these signal domains:

- spectrum to spectrum

6.1.0.2 Categories

feedback

6.1.0.3 Configuration variables

Name	Type	Description	Default
mhaconfig_in	parser	Input configuration	(see below)
mhaconfig_out	parser	Output configuration	(see below)
df	vector<float>	frequency to shift the bins / Hz	[40]
fmin	float	lower boundary for frequency shifter **Range:** [0,]	4000
fmax	float	upper boundary for frequency shifter **Range:** [0,]	16000
irslen	int	Bandpass: maximum length of cut off filter response **Range:** [1,]	1
phasemode	keyword_list	Bandpass: mode of gain smoothing **Range:** [none linear minimal]	none

Variables of sub-parser `mhaconfig_in`:

Name	Type	Description	Default
channels	int	Number of audio channels	(monitor)
domain	string	Signal domain (MHA_WAVEFORM or MHA_SPECTRUM)	(monitor)
fragsize	int	Fragment size of waveform data	(monitor)
wndlen	int	Window length of spectral data	(monitor)
fftlen	int	FFT length of spectral data	(monitor)
srate	float	Sampling rate in Hz	(monitor)

Variables of sub-parser `mhaconfig_out`:

Name	Type	Description	Default
channels	int	Number of audio channels	(monitor)
domain	string	Signal domain (MHA_WAVEFORM or MHA_SPECTRUM)	(monitor)
fragsize	int	Fragment size of waveform data	(monitor)
wndlen	int	Window length of spectral data	(monitor)
fftlen	int	FFT length of spectral data	(monitor)
srate	float	Sampling rate in Hz	(monitor)

7 Plugin category 'filter'

7.1 combinechannels

Channel combiner

7.1.0.1 Detailed description

Several filter channels can be combined into one or more outputchannels by summing the input channels. This plugin is intended as afilter resynthesis of linear-phase filter banks.

The input signalis expected to have a non-interleaved channel order, i.e., first allbands of first output channel, then all bands of second channel, etc.

7.1.0.2 Supported domains

The MHA plugin `combinechannels` supportes these signal domains:

- waveform to waveform
- spectrum to spectrum

7.1.0.3 Categories

filter

7.1.0.4 Configuration variables

Name	Type	Description	Default
mhaconfig_in	parser	Input configuration	(see below)
mhaconfig_out	parser	Output configuration	(see below)
outchannels	int	Number of output channels **Range:** [1,]	1
interleaved	bool	Input signal has interleaved channel order?	no
channel_gain_name	string	Name of channel gain AC variable (looked up during prepare, can be empty)	
element_gain_name	string	Name of element wise gain AC variable (looked up during waveform process, can be empty)	

Variables of sub-parser `mhaconfig_in`:

Name	Type	Description	Default
channels	int	Number of audio channels	(monitor)
domain	string	Signal domain (MHA_WAVEFORM or MHA_SPECTRUM)	(monitor)
fragsize	int	Fragment size of waveform data	(monitor)
wndlen	int	Window length of spectral data	(monitor)
fftlen	int	FFT length of spectral data	(monitor)
srate	float	Sampling rate in Hz	(monitor)

Variables of sub-parser `mhaconfig_out`:

Name	Type	Description	Default
channels	int	Number of audio channels	(monitor)
domain	string	Signal domain (MHA_WAVEFORM or MHA_SPECTRUM)	(monitor)
fragsize	int	Fragment size of waveform data	(monitor)
wndlen	int	Window length of spectral data	(monitor)
fftlen	int	FFT length of spectral data	(monitor)
srate	float	Sampling rate in Hz	(monitor)

7.2 fftfilterbank

FFT based filterbank with overlapping filters

7.2.0.1 Detailed description

This plugin implements a linear phase filterbank based on FFT spectrum. Each filter bank channel is stored into an own audio channel. The number of output channels of this plugin is the number of frequency bands times the number of input channels.

Please use the iFFT plugin *spec2wave* (p. **??**) to get the waveform signal of the filterbank output. The *matrixmixer* (p. **??**) plugin or *combinechannels* (p. 17) can be used for resynthesis.

The filters are calculated by applying filter weights to each FFT bin. These weights (filter shapes) depend on the settings of the `use_edges` variable. If `center` is selected, the frequency interval between the lower neighbour center frequency and the desired center frequency is mapped to the interval [-1,0] and between the desired center frequency and the upper neightbour to the interval [0,1]. These mappings are linear on the given frequency scale so that a value of 0.5 denotes the middle between two neighboured center frequencies on the given frequency scale. The filter weights are calculated with the configured crossing function on this interval, see next figure for details. Please note that the filters are not necessarily symmetric (symmetry is achieved only if the center frequencies are equally spaced on the desired frequency scale). The lowest and highest filter channels include the full range from zero to the center frequency or from the center frequency to the nyquist frequency, respectively.

If `edge` is selected, then the frequency axis is transformed to be linear on the desired frequency scale. The interval between two edge frequencies is mapped to [-0.5,0.5]. Now, the filter shape function (rectangular, linear/sawtooth, hanning) is applied to the frequency axis. This results in symmetric filters on the desired frequency scale.

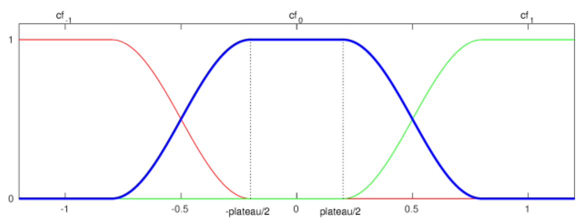

Figure 1 Schematic plot of overlapping filters

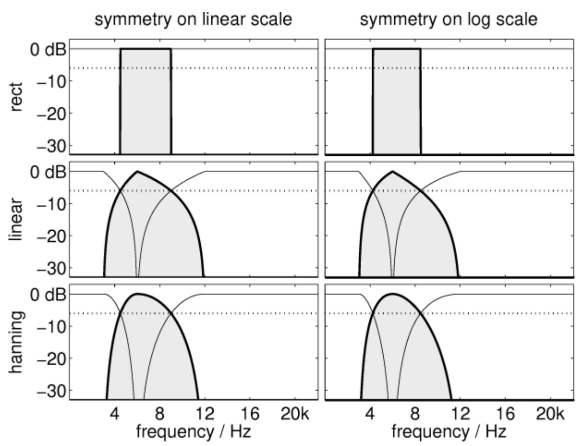

Figure 2 Example filter shapes with center frequencies configured

7.2.0.2 Supported domains

The MHA plugin `fftfilterbank` supportes these signal domains:

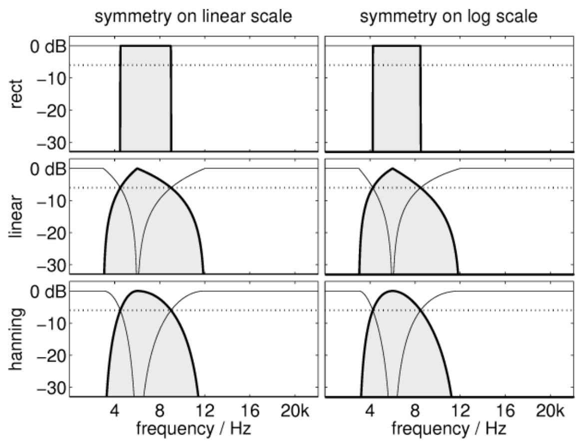

Figure 3 Example filter shapes with edge frequencies configured

- waveform to waveform

- spectrum to spectrum

7.2.0.3 Categories

filter

7.2.0.4 Configuration variables

Name	Type	Description	Default
mhaconfig_in	parser	Input configuration	(see below)
mhaconfig_out	parser	Output configuration	(see below)
unit	keyword_list	Frequency unit **Range:** [Hz kHz Oct Oct/3 Bark Erb ERB_Glasberg1990]	Hz
f	vector<float>	Frequencies	[]
f_hz	vector<float>	Frequencies in Hz	(monitor)
fscale	keyword_list	frequency scale of filter bank **Range:** [linear bark log erb ERB_Glasberg1990]	linear
ovltype	keyword_list	filter overlap type **Range:** [rect linear hanning exp gauss]	rect
plateau	float	relative plateau width **Range:** [0,1[	0
ftype	keyword_list	frequency entry type **Range:** [center edge]	center
normalize	bool	normalize broadband output amplitude	no
fail_on_nonmonotonic	bool	Fail if frequency entries are non-monotonic (otherwise sort)	yes
fail_on_unique_bins	bool	Fail if center frequencies share the same FFT bin.	yes
cf	vector<float>	final center frequencies in Hz	(monitor)
ef	vector<float>	final edge frequencies in Hz	(monitor)
cLTASS	vector<float>	Bandwidth level correction for LTASS noise in dB	(monitor)
shapes	matrix<float>	Frequency band shapes	(monitor)
fftlen	int	FFT length of filterbank (affects time domain only) **Range:** [2,]	128
phasemodel	keyword_list	Phase model (affects time domain only) **Range:** [minimal linear]	linear
irswnd	parser	IRS window function (affects time domain only)	(see below)
return_imag	bool	Return imaginary part? Results are stored in AC variable '<plugname>_imag'.	no

Variables of sub-parser `mhaconfig_in`:

Name	Type	Description	Default
channels	int	Number of audio channels	(monitor)
domain	string	Signal domain (MHA_WAVEFORM or MHA_SPECTRUM)	(monitor)
fragsize	int	Fragment size of waveform data	(monitor)
wndlen	int	Window length of spectral data	(monitor)
fftlen	int	FFT length of spectral data	(monitor)
srate	float	Sampling rate in Hz	(monitor)

Variables of sub-parser `mhaconfig_out`:

Name	Type	Description	Default
channels	int	Number of audio channels	(monitor)
domain	string	Signal domain (MHA_WAVEFORM or MHA_SPECTRUM)	(monitor)
fragsize	int	Fragment size of waveform data	(monitor)
wndlen	int	Window length of spectral data	(monitor)
fftlen	int	FFT length of spectral data	(monitor)
srate	float	Sampling rate in Hz	(monitor)

Variables of sub-parser `irswnd`:

Name	Type	Description	Default
type	keyword_list	Window type. **Range:** [rect hanning hamming blackman bartlett user]	hanning
user	vector<float>	User provided window (used if window type==user).	[]

8 Plugin category 'level'

8.1 dc

dynamic compression

8.1.0.1 Supported domains

The MHA plugin `dc` supportes these signal domains:

- waveform to waveform
- spectrum to spectrum

8.1.0.2 Categories

level compression

8.1.0.3 Configuration variables

Name	Type	Description	Default
mhaconfig_in	parser	Input configuration	(see below)
mhaconfig_out	parser	Output configuration	(see below)
powersum	bool	Input level is summed accross channels	no
gtdata	matrix<float>	gaintable data in dB gains	[[]]
gtmin	vector<float>	input level for first gain entry in dB SPL	[]
gtstep	vector<float>	level step size in dB	[]
tau_rmslev	vector<float>	RMS level averaging time constant in s	[]
tau_attack	vector<float>	attack time constant in s	[]
tau_decay	vector<float>	decay time constant in s	[]
fb	string	Name of fftfilterbank plugin. Used to extract frequency information.	fftfilterbank
chname	string	name of audio channel number variable (empty: broadband)	
bypass	bool	bypass dynamic compression	no
clientid	string	Client ID of last fit	
gainrule	string	Gain rule of last fit	
preset	string	Preset name of last fit	
modified	int	Flag if configuration has been modified	(monitor)
max_level_difference	matrix<float>	maximum level difference in dB between adjacent bands **Range:** [0,[	[[]]
level_in	vector<float>	input level of last block / dB SPL	(monitor)
level_in_filtered	vector<float>	input level after time-constant filters / dB SPL	(monitor)
cf	vector<float>	nominal center frequencies of filterbank bands	(monitor)
ef	vector<float>	edge frequencies of filterbank bands	(monitor)
band_weights	vector<float>	Weights of the individual frequency bands. Computed as (sum of squared fft-bin-weigths) / num_frames.	(monitor)
use_wbinhib	bool	Use wideband inhibition?	no
wbinhib	parser		(see below)

Variables of sub-parser `mhaconfig_in`:

Name	Type	Description	Default
channels	int	Number of audio channels	(monitor)
domain	string	Signal domain (MHA_WAVEFORM or MHA_SPECTRUM)	(monitor)
fragsize	int	Fragment size of waveform data	(monitor)
wndlen	int	Window length of spectral data	(monitor)
fftlen	int	FFT length of spectral data	(monitor)
srate	float	Sampling rate in Hz	(monitor)

Variables of sub-parser `mhaconfig_out`:

Name	Type	Description	Default
channels	int	Number of audio channels	(monitor)
domain	string	Signal domain (MHA_WAVEFORM or MHA_SPECTRUM)	(monitor)
fragsize	int	Fragment size of waveform data	(monitor)
wndlen	int	Window length of spectral data	(monitor)
fftlen	int	FFT length of spectral data	(monitor)
srate	float	Sampling rate in Hz	(monitor)

Variables of sub-parser `wbinhib`:

Name	Type	Description	Default
weights	vector<float>	weighting of neighbour frequencies	[1 2 4 4 4 2 1]
dl_map_min	float	mapping of level/broadband level ratio (in dB) into inhibition ratio, lower value means full inhibition	-8
dl_map_max	float	mapping of level/broadband level ratio (in dB) into inhibition ratio, upper value means no inhibition	2
l_min	float	level threshold, inhibition starts above this level	40
g_scale	matrix<float>	scaling factor of inhibition gain	[[1]]

8.2 transducers

Signal level calibration plugin.

8.2.0.1 Detailed description

Some plugins in the MHA expect the input signal to be calibrated to sound pressure level in Pascal. This plugin converts AD and DA converter levels to SPL in Pa and also allows for a FIR filters for mircophone and receiver equalization.

A schematic calibration rule for the MHA

1. Measure frequency response of hearing aid microphones and receiver.
2. Create FIR filter coefficients for frequency response equalization for microphones and receiver, configure the FIR coefficients of this plugin correspondingly.
3. Play an acoustic reference signal of a known SPL level to the microphone, adjust the 'calib_in.peaklevel' variable until the internal level meter (e.g. rmslevel, p. **??**) shows the same level.
4. Create a test tone in the MHA (e.g. with 'noise', p. **??**, or 'sine', p. **??**) of a given level, and adjust the variable 'calib_out.peaklevel' until the same acoustic level is measured at the receiver.

Besides the signal calibration, this plugin also contains a soft-limiter in the output path, and a quantization module. The soft-limiter acts as a fast broadband compressor, and can be configured correspondingly. The quantisation module limits the signal to the interval $[-1, 1]$ and optionally reduces the resolution, by this quantization rule:

$$y = \mathrm{floor}(2^{(N-1)}x)2^{-(N-1)} \tag{2}$$

N is the number of bits, x the input signal and y the output signal.

8.2.0.2 Supported domains

The MHA plugin `transducers` supportes these signal domains:

- waveform to waveform

8.2.0.3 Categories

level

8.2.0.4 Configuration variables

Name	Type	Description	Default
mhaconfig_in	parser	Input configuration	(see below)
mhaconfig_out	parser	Output configuration	(see below)
plugin_name	string	Plugin name	
calib_in	parser	calibration module	(see below)
calib_out	parser	calibration module	(see below)

Variables of sub-parser `mhaconfig_in`:

Name	Type	Description	Default
channels	int	Number of audio channels	(monitor)
domain	string	Signal domain (MHA_WAVEFORM or MHA_SPECTRUM)	(monitor)
fragsize	int	Fragment size of waveform data	(monitor)
wndlen	int	Window length of spectral data	(monitor)
fftlen	int	FFT length of spectral data	(monitor)
srate	float	Sampling rate in Hz	(monitor)

Variables of sub-parser `mhaconfig_out`:

Name	Type	Description	Default
channels	int	Number of audio channels	(monitor)
domain	string	Signal domain (MHA_WAVEFORM or MHA_SPECTRUM)	(monitor)
fragsize	int	Fragment size of waveform data	(monitor)
wndlen	int	Window length of spectral data	(monitor)
fftlen	int	FFT length of spectral data	(monitor)
srate	float	Sampling rate in Hz	(monitor)

Variables of sub-parser `calib_in`:

Name	Type	Description	Default
mhaconfig_in	parser	Input configuration	(see below)
mhaconfig_out	parser	Output configuration	(see below)
nbits	int	Number of bits to simulate, or zero for limiting only **Range:** [0,32]	0
fir	matrix<float>	FIR filter coefficients, one row for each channel	[[]]
peaklevel	vector<float>	Reference peak level in dB (0 dB FS corresponds to this SPL level)	[]
speechnoise	parser		(see below)
tau_level	float	Time constant in seconds for RMS level meter **Range:**]0,10]	0.125
rmslevel	vector<float>	RMS level in dB at input (after calibration or addition of noise)	(monitor)
config	parser		(see below)

Variables of sub-parser `mhaconfig_in`:

Name	Type	Description	Default
channels	int	Number of audio channels	(monitor)
domain	string	Signal domain (MHA_WAVEFORM or MHA_SPECTRUM)	(monitor)
fragsize	int	Fragment size of waveform data	(monitor)
wndlen	int	Window length of spectral data	(monitor)
fftlen	int	FFT length of spectral data	(monitor)
srate	float	Sampling rate in Hz	(monitor)

Variables of sub-parser `mhaconfig_out`:

Name	Type	Description	Default
channels	int	Number of audio channels	(monitor)
domain	string	Signal domain (MHA_WAVEFORM or MHA_SPECTRUM)	(monitor)
fragsize	int	Fragment size of waveform data	(monitor)
wndlen	int	Window length of spectral data	(monitor)
fftlen	int	FFT length of spectral data	(monitor)
srate	float	Sampling rate in Hz	(monitor)

Variables of sub-parser `speechnoise`:

Name	Type	Description	Default
mode	keyword_list	Playback mode and level of speech shaped noise **Range:** [off on olnoise LTASS_combined LTASS_female LTASS_male white pink brown TEN_SPL TEN_SPL_250_8k TEN_SPL_50_16k sin125 sin250 sin500 sin1k sin2k sin4k sin8k]	off
level	float	Test signal level in dB SPL **Range:** [0,120]	80
channels	vector<int>	Channels where to playb speech noise signal **Range:** [0,]	[]

Variables of sub-parser `config`:

Name	Type	Description	Default
srate	float	Actual sampling rate / Hz	(monitor)
fragsize	int	Actual fragment size / samples	(monitor)
channels	int	Actual number of channels	(monitor)

Variables of sub-parser `calib_out`:

Name	Type	Description	Default
mhaconfig_in	parser	Input configuration	(see below)
mhaconfig_out	parser	Output configuration	(see below)
speechnoise	parser		(see below)
peaklevel	vector<float>	Reference peak level in dB (0 dB FS corresponds to this SPL level)	[]
fir	matrix<float>	FIR filter coefficients, one row for each channel	[[]]
softclip	parser	'Hardware' softclipper	(see below)
nbits	int	Number of bits to simulate, or zero for limiting only **Range:** [0,32]	0
do_clipping	bool	Will the soft/ hard clipping be executed	no
tau_level	float	Time constant in seconds for RMS level meter **Range:**]0,10]	0.125
rmslevel	vector<float>	RMS level in dB at output (before calibration or addition of noise)	(monitor)
config	parser		(see below)

Variables of sub-parser `mhaconfig_in`:

Name	Type	Description	Default
channels	int	Number of audio channels	(monitor)
domain	string	Signal domain (MHA_WAVEFORM or MHA_SPECTRUM)	(monitor)
fragsize	int	Fragment size of waveform data	(monitor)
wndlen	int	Window length of spectral data	(monitor)
fftlen	int	FFT length of spectral data	(monitor)
srate	float	Sampling rate in Hz	(monitor)

Variables of sub-parser `mhaconfig_out`:

Name	Type	Description	Default
channels	int	Number of audio channels	(monitor)
domain	string	Signal domain (MHA_WAVEFORM or MHA_SPECTRUM)	(monitor)
fragsize	int	Fragment size of waveform data	(monitor)
wndlen	int	Window length of spectral data	(monitor)
fftlen	int	FFT length of spectral data	(monitor)
srate	float	Sampling rate in Hz	(monitor)

Variables of sub-parser `speechnoise`:

Name	Type	Description	Default
mode	keyword_list	Playback mode and level of speech shaped noise **Range:** [off on olnoise LTASS_combined LTASS_female LTASS_male white pink brown TEN_SPL TEN_SPL_250_8k TEN_SPL_50_16k sin125 sin250 sin500 sin1k sin2k sin4k sin8k]	off
level	float	Test signal level in dB SPL **Range:** [0,120]	80
channels	vector<int>	Channels where to playb speech noise signal **Range:** [0,]	[]

Variables of sub-parser `softclip`:

Name	Type	Description	Default
tau_attack	float	attack filter time constant / s **Range:** [0,]	0.002
tau_decay	float	decay filter time constant / s **Range:** [0,]	0.005
threshold	float	start point on linear scale (hard clipping at 1.0) **Range:** [0,]	0.6
slope	float	compression factor **Range:** [0,1]	0.5
linear	bool	input/output function is linear on linear (yes) or logarithmic (no) scale	no
tau_clip	float	clipping meter time constant / s **Range:** [0,]	1
clipped	float	clipped ratio	(monitor)
max_clipped	float	maximum allowed clipped ratio **Range:** [0,1]	1

Variables of sub-parser `config`:

Name	Type	Description	Default
srate	float	Actual sampling rate / Hz	(monitor)
fragsize	int	Actual fragment size / samples	(monitor)
channels	int	Actual number of channels	(monitor)

9 Plugin category 'noise'

9.1 timoSmooth

Cepstral smoothing single-channel noise reduction

9.1.0.1 Detailed description

Implements the single-channel noise reduction scheme found in Breithaupt, Gerkmann, and Martin, A Novel A Priori SNR Estimation Approach Based on Selective Cepstro-temporal Smoothing.

9.1.0.2 Supported domains

The MHA plugin `timoSmooth` supportes these signal domains:

- spectrum to spectrum

9.1.0.3 Categories

noise reduction

9.1.0.4 Configuration variables

Name	Type	Description	Default
mhaconfig_in	parser	Input configuration	(see below)
mhaconfig_out	parser	Output configuration	(see below)
xi_min_db	float	Minimum a priori SNR for a bin in dB(power) **Range:** [-50,50]	-27
f0_low	float	Lower limit for F0 detection in Hz **Range:** [0,400]	70
f0_high	float	Upper limit for F0 detection in Hz **Range:** [0,400]	300
delta_pitch	float	Quefrency half-width of pitch-set in samps **Range:** [0,20]	2
lambda_thresh	float	Pitch detection threshold for smooth cepstrum in magnitude **Range:** [0,3]	0.2
alpha_pitch	float	Alpha value to set for pitch range **Range:** [0,4]	0.15
beta_const	float	AR coeff for smoothing of alphas(smoothing-factors)	0.96
kappa_const	float	Exponential bias correction constant for a priori SNR estimate **Range:** [0,1]	0.2886
gain_min_db	float	Minimum gain in dB for a frequency bin **Range:** [-30,0]	-17
win_f0	vector<float>	Window coefficients for cepstral smoothing window **Range:** [0,1]	[0.0207 0.0656 0.1664 0.2473 0.24
alpha_const_vals	vector<float>	Piecewise values for steady-state alphas **Range:** [0,2]	[0.2 0.4 0.92]
alpha_const_limits_hz	vector<float>	Limits for steady-state alphas given in Hz **Range:** [0,10000]	[93.75 625]
noisePow_name	string	Name of est. noise spectrum in AC space	noisePowProposedScale
spp	parser	Subparser for exporting SPP	(see below)

Variables of sub-parser `mhaconfig_in`:

Name	Type	Description	Default
channels	int	Number of audio channels	(monitor)
domain	string	Signal domain (MHA_WAVEFORM or MHA_SPECTRUM)	(monitor)
fragsize	int	Fragment size of waveform data	(monitor)
wndlen	int	Window length of spectral data	(monitor)
fftlen	int	FFT length of spectral data	(monitor)
srate	float	Sampling rate in Hz	(monitor)

Variables of sub-parser `mhaconfig_out`:

Name	Type	Description	Default
channels	int	Number of audio channels	(monitor)
domain	string	Signal domain (MHA_WAVEFORM or MHA_SPECTRUM)	(monitor)
fragsize	int	Fragment size of waveform data	(monitor)
wndlen	int	Window length of spectral data	(monitor)
fftlen	int	FFT length of spectral data	(monitor)
srate	float	Sampling rate in Hz	(monitor)

Variables of sub-parser `spp`:

Name	Type	Description	Default
prior_q	float	priorQ for computing GLR and SPP from local SNR **Range:** [0,2]	0.5
xi_opt_db	float	xiOpt in dB for computing GLR and SPP from local SNR **Range:** [0,40]	15

10 Plugin category 'overlapadd'

10.1 overlapadd

Waveform to spectrum overlap add and FFT method. Audio data is collected up to wndlen, than windowed with the given window function, zero padded up to fftlength (symmetric zero padding or asymmetric zero padding possible), and Fast-Fourier-transformed. All parameter changes take effect after the next prepare call.

10.1.0.1 Detailed description

This plugin transforms fragmented waveform data into short time Fourier transformed audio (STFT), and after processing by spectral processing plugins back to the time domain. This overlap-add mechanism is similar to that from Allen (1977): First, the waveform signal is windowed with a window function, e.g., a Hanning window. In each processing frame, the window is shifted by the fragment size of the input waveform. Missing parts of the signal are taken from the past. The windowed signal is padded with zeros on both sides up to the FFT length, to avoid aliasing when filters are applied in the frequency domain. The impulse response of the applied filter can have the length of the zero padding; if the impulse response is longer, later parts of the impulse response will be mapped to the beginning of the fragment (temporal aliasing). Linear phase filters (real gains in the frequency domain) produce symmetric impulse responses and therefore require symmetric zero padding. The zero padded signal is now fast Fourier transformed. Parameters are FFT length N, window length M and the fragment size P. Typical values for the window length are $M = 2P$ or $M = 4P$. The Hanning window used in the first step is $w_1(k) = \frac{1}{2}(1 - \cos(2\pi k/M))$, the windowed signal is

$$x_w(m, k) = w_1(k) \cdot x(m \cdot P + k), \tag{3}$$

with $k = 0, \ldots, M - 1$ and the fragment index m.

After processing and inverse Fourier transformation, Hanning ramps are applied to the signal to avoid discontinuities in case of temporal aliasing, and thus reducing the artifacts. The length of the Hanning ramps are a fraction p of half the zero-padding length $(N-M)/2$. If $p=1$, the entire zero-padded parts are smoothed with Hanning ramps. $p=0$ means, that no Hanning ramps are applied to the signal. This allows an exact reproduction in those cases, where the local impulse response of the filter (represented by all algorithms between FFT and inverse FFT) is shorter than the zero padding length. The windowing in both stages of the overlap-add mechanism is plotted in Fig. 4 for $M=2P$ (50

The total delay between input and output of a real-time system with fragment size P and an overlap-add based linear-phase filter, is the window length plus half the zero-padding length, or $M+(N-M)/2$, plus an additional delay needed for the signal processing, and plus a delay generated by the AD/DA converters (e.g., anti-aliasing filter delay). In an offline system, the complete input signal is available in advance, and thus the delay of the overlap-add method is determined only by the relative shift between output and input signal, which is $(M+N)/2-P$ (equal to $N/2$ in case of 50% overlap, i.e. $M=2P$). Contrary to a real-time system, the delay of an offline system depends on the amount of overlap.

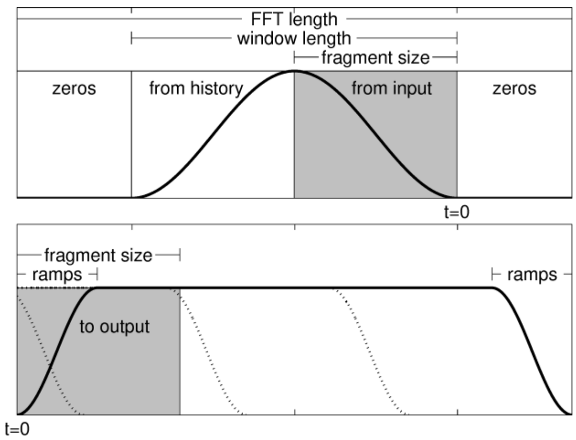

Figure 4 Windowing in the overlap-add method with 50% overlap and zero-padding. In the upper panel, the windowed input signal before applying the FFT is schematically plotted. In the lower panel, the same time interval after inverse FFT is shown. The shaded segment is the fragment which is read from the input stream (upper panel) and written to the output stream (lower panel) in one processing cycle. The delay between input and output signal is the length of leading zeros plus the window length.

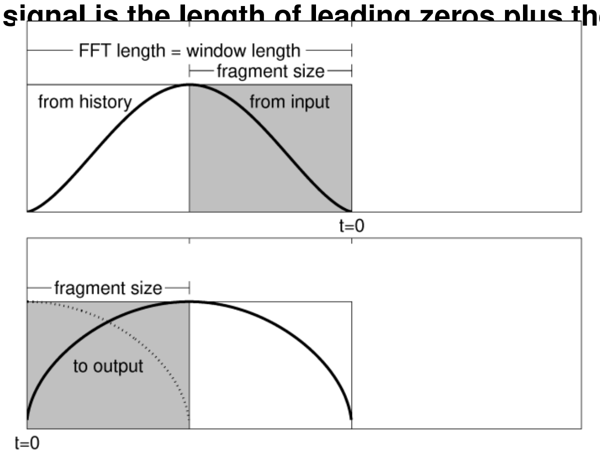

Figure 5 Windowing in the overlap-add method, as in Fig. 4, but with post-windowing and without zero-padding. In this setup, W^{α} is applied before FFT and $W^{1-\alpha}$ is used for post-windowing. The delay between input and output signal is the window length.

10.1.0.2 Supported domains

The MHA plugin `overlapadd` supportes these signal domains:

- waveform to waveform

10.1.0.3 Categories

overlapadd

10.1.0.4 Configuration variables

Name	Type	Description	Default
mhaconfig_in	parser	Input configuration	(see below)
mhaconfig_out	parser	Output configuration	(see below)
plugin_name	string	Plugin name	
fftlen	int	FFT length **Range:** [1,]	512
wnd	parser	window type	(see below)
zerownd	parser	zero padding post window type	(see below)
prescale	float	scaling factor (pre-scaling)	(monitor)
postscale	float	scaling factor (post-scaling)	(monitor)

Variables of sub-parser `mhaconfig_in`:

Name	Type	Description	Default
channels	int	Number of audio channels	(monitor)
domain	string	Signal domain (MHA_WAVEFORM or MHA_SPECTRUM)	(monitor)
fragsize	int	Fragment size of waveform data	(monitor)
wndlen	int	Window length of spectral data	(monitor)
fftlen	int	FFT length of spectral data	(monitor)
srate	float	Sampling rate in Hz	(monitor)

Variables of sub-parser `mhaconfig_out`:

Name	Type	Description	Default
channels	int	Number of audio channels	(monitor)
domain	string	Signal domain (MHA_WAVEFORM or MHA_SPECTRUM)	(monitor)
fragsize	int	Fragment size of waveform data	(monitor)
wndlen	int	Window length of spectral data	(monitor)
fftlen	int	FFT length of spectral data	(monitor)
srate	float	Sampling rate in Hz	(monitor)

Variables of sub-parser `wnd`:

Name	Type	Description	Default
type	keyword_list	Window type. **Range:** [rect hanning hamming blackman bartlett user]	hanning
user	vector<float>	User provided window (used if window type==user).	[]
len	int	window length/samples **Range:** [1,]	400
pos	float	window position (0 = beginning, 0.5 = symmetric zero padding, 1 = end) **Range:** [0,1]	0.5
exp	float	window exponent to be applied to all elements of window function	1

Variables of sub-parser `zerownd`:

Name	Type	Description	Default
type	keyword_list	Window type. **Range:** [rect hanning hamming blackman bartlett user]	rect
user	vector<float>	User provided window (used if window type==user).	[]

11 Plugin category 'signalflow'

11.1 db

Synchronous double buffer plugin.

11.1.0.1 Detailed description

The double buffer plugin allows changes of fragment size. It has an outer layer (e.g. framework) and an inner layer (e.g. MHA kernel, plugin). A configurable fragment size is used on the inner side, which is independent from the outer fragment size. The input data is buffered, and the data is processed when enough samples are available. The configuration of the inner plugin is available via the `plug` prefix.

Please note that double buffering adds an extra delay of the audio stream. If both fragment sizes are identical, the double buffering is bypassed.

11.1.0.2 Warning:

If the inner fragment size is larger than the outer fragment size, the maximal processing time is limited by the shorter fragment size. This results in a maximal processor usage determined by the ratio of outer to inner fragment size. This problem holds not for offline processing. As an alternative, the asynchronous double-buffer plugin `dbasync` (section **??**) can be used, which processes the double-buffered signal in a separate thread. That plugin should be preferred for real-time processing. If the inner thread should only be used for signal analysis, please refer to the plugin `analysispath` (section **??**).

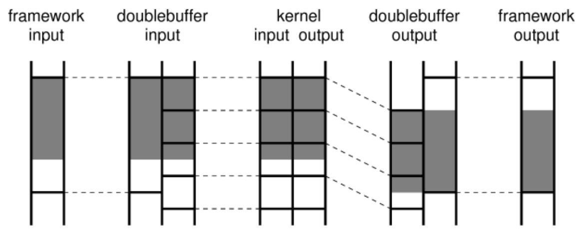

Figure 6 Concept of the double buffer plugin. The outer fragments, provided by the framework, are split up into smaller fragments for processing in the kernel. For a continuous output stream, an extra delay is needed, i.e. the first fragment is filled with zeros at the beginning.

11.1.0.3 Supported domains

The MHA plugin `db` supportes these signal domains:

- waveform to waveform

11.1.0.4 Categories

signalflow

11.1.0.5 Configuration variables

Name	Type	Description	Default
mhaconfig_in	parser	Input configuration	(see below)
mhaconfig_out	parser	Output configuration	(see below)
plugin_name	string	Plugin name	
fragsize	int	fragment size of client plugin **Range:** [0,]	200

Variables of sub-parser `mhaconfig_in`:

Name	Type	Description	Default
channels	int	Number of audio channels	(monitor)
domain	string	Signal domain (MHA_WAVEFORM or MHA_SPECTRUM)	(monitor)
fragsize	int	Fragment size of waveform data	(monitor)
wndlen	int	Window length of spectral data	(monitor)
fftlen	int	FFT length of spectral data	(monitor)
srate	float	Sampling rate in Hz	(monitor)

Variables of sub-parser `mhaconfig_out`:

Name	Type	Description	Default
channels	int	Number of audio channels	(monitor)
domain	string	Signal domain (MHA_WAVEFORM or MHA_SPECTRUM)	(monitor)
fragsize	int	Fragment size of waveform data	(monitor)
wndlen	int	Window length of spectral data	(monitor)
fftlen	int	FFT length of spectral data	(monitor)
srate	float	Sampling rate in Hz	(monitor)

11.2 mhachain

MHA Chain

11.2.0.1 Detailed description

Load a sequence of plugins. During processing, the signal is passed from plugin to plugin, and may change its domain or dimension.

If profiling is switched on, the cumulative time spent in the processing callback of each plugin is stored in a monitor variable.

The complete chain can be replaced by other algorithms during run-time if the default configuration is valid for processing and if the output domain or dimension does not change by replacing the chain.

11.2.0.2 Supported domains

The MHA plugin `mhachain` supportes these signal domains:

- waveform to waveform
- waveform to spectrum
- spectrum to waveform
- spectrum to spectrum

11.2.0.3 Categories

signalflow

11.2.0.4 Configuration variables

Name	Type	Description	Default
mhaconfig_in	parser	Input configuration	(see below)
mhaconfig_out	parser	Output configuration	(see below)
use_profiling	bool	use profiling method	no
algos	vector<string>	list of plugins	[]

Variables of sub-parser `mhaconfig_in`:

Name	Type	Description	Default
channels	int	Number of audio channels	(monitor)
domain	string	Signal domain (MHA_WAVEFORM or MHA_SPECTRUM)	(monitor)
fragsize	int	Fragment size of waveform data	(monitor)
wndlen	int	Window length of spectral data	(monitor)
fftlen	int	FFT length of spectral data	(monitor)
srate	float	Sampling rate in Hz	(monitor)

Variables of sub-parser `mhaconfig_out`:

Name	Type	Description	Default
channels	int	Number of audio channels	(monitor)
domain	string	Signal domain (MHA_WAVEFORM or MHA_SPECTRUM)	(monitor)
fragsize	int	Fragment size of waveform data	(monitor)
wndlen	int	Window length of spectral data	(monitor)
fftlen	int	FFT length of spectral data	(monitor)
srate	float	Sampling rate in Hz	(monitor)

11.3 route

Signal router plugin.

Arguments are the input signal source names (AC variables) followed by a colon, followed by the channel number, starting at zero. Empty names correspond to the direct input.

An AC variable will be created if the AC output dimension is not zero. Example: out = [:0 :1 x:0 x:1] ac = [:2 :3] returns a four channel output signal containing first two direct input channels, and the first two channels of the AC variable "x". An AC variable is created with the third and fourth channel of the direct input.

11.3.0.1 Supported domains

The MHA plugin `route` supportes these signal domains:

- waveform to waveform
- spectrum to spectrum

11.3.0.2 Categories

signalflow

11.3.0.3 Configuration variables

Name	Type	Description	Default
mhaconfig_in	parser	Input configuration	(see below)
mhaconfig_out	parser	Output configuration	(see below)
out	vector<string>	direct output	[]
ac	vector<string>	AC output	[]

Variables of sub-parser `mhaconfig_in`:

Name	Type	Description	Default
channels	int	Number of audio channels	(monitor)
domain	string	Signal domain (MHA_WAVEFORM or MHA_SPECTRUM)	(monitor)
fragsize	int	Fragment size of waveform data	(monitor)
wndlen	int	Window length of spectral data	(monitor)
fftlen	int	FFT length of spectral data	(monitor)
srate	float	Sampling rate in Hz	(monitor)

Variables of sub-parser `mhaconfig_out`:

Name	Type	Description	Default
channels	int	Number of audio channels	(monitor)
domain	string	Signal domain (MHA_WAVEFORM or MHA_SPECTRUM)	(monitor)
fragsize	int	Fragment size of waveform data	(monitor)
wndlen	int	Window length of spectral data	(monitor)
fftlen	int	FFT length of spectral data	(monitor)
srate	float	Sampling rate in Hz	(monitor)

12 Uncategorized plugins

12.1 noisePowProposedScale

noise power estimator after Gerkmann (2012)

Copyright © 2011 Author: Timo Gerkmann and Richard Hendriks Royal Institute of Technology (KTH) Delft university of Technology

12.1.0.1 Supported domains

The MHA plugin `noisePowProposedScale` supportes these signal domains:

- spectrum to spectrum

12.1.0.2 Categories

other

12.1.0.3 Configuration variables

Name	Type	Description	Default
mhaconfig_in	parser	Input configuration	(see below)
mhaconfig_out	parser	Output configuration	(see below)
alphaPH1mean	float	low pass filter coefficient for PH1mean **Range:** [0,1[	0.9
alphaPSD	float	low pass filter coefficient for PSD **Range:** [0,1[	0.8
q	float	a priori probability of speech presence **Range:** [0,1]	0.5
xiOptDb	float	optimal fixed a priori SNR for SPP estimation	15

Variables of sub-parser `mhaconfig_in`:

Name	Type	Description	Default
channels	int	Number of audio channels	(monitor)
domain	string	Signal domain (MHA_WAVEFORM or MHA_SPECTRUM)	(monitor)
fragsize	int	Fragment size of waveform data	(monitor)
wndlen	int	Window length of spectral data	(monitor)
fftlen	int	FFT length of spectral data	(monitor)
srate	float	Sampling rate in Hz	(monitor)

Variables of sub-parser `mhaconfig_out`:

Name	Type	Description	Default
channels	int	Number of audio channels	(monitor)
domain	string	Signal domain (MHA_WAVEFORM or MHA_SPECTRUM)	(monitor)
fragsize	int	Fragment size of waveform data	(monitor)
wndlen	int	Window length of spectral data	(monitor)
fftlen	int	FFT length of spectral data	(monitor)
srate	float	Sampling rate in Hz	(monitor)

References

Index