CHAPRO

Compression Hearing-Aid Processing Library
API Documentation

Boys Town National Research Hospital
25 November 2016

TABLE OF CONTENTS

CHAPRO LIBRARY OVERVIEW

CHAPRO is a library of functions that may be used to implement simulations of compression hearing-aid signal processing. Two different types of signal processing strategies are included: (1) gammatone filter-bank frequency analysis with instantaneous compression and (2) FIR filter-bank frequency analysis with automatic gain control.

A modular design has been adopted to facilitate replacement of library function with alternative signal-processing implementations. Each of the five major modules contains (1) a preparation function that allocates memory and initializes variables and (2) one or two processing functions that perform signal processing.

1. Complex Gammatone filter-bank
   a. `cgtfb_prepare`
   b. `cgtfb_analyze`
   c. `cgtfb_synthesize`
2. Instantaneous compression
   a. `compressor_prepare`
   b. `compressor_process`
3. FIR filter-bank Fourier Transform
   a. `firfb_prepare`
   b. `firfb_analyze`
   c. `firfb_synthesize`
4. Automatic gain control
   a. `agc_prepare`
   b. `agc_input`
   c. `agc_channel`
   d. `agc_output`
5. Feedback management
   a. `feedback_prepare`
   b. `feedback_manage`
   c. `feedback_record`

All variables initialized and data memory allocated by the preparation functions are combined into a single data structure to facilitate the creation of firmware for real-time implementation on signal-processing hardware. The CHAPRO library includes a function that generates a C-code representation of this initialized data.

- `data_gen`

For desktop simulation, the CHAPRO library includes core functions for memory allocation and disposal.

- `prepare`
- `allocate`

Functions for FFT of real signals are included among the core functions.

- `fft_rc`

- `fft_cr`
- `cleanup`

Finally, the CHAPRO library includes a function that returns a version description string.

- `version`

To simulate gammatone filter-bank frequency analysis with instantaneous compression, variable initialization and memory allocation is performed by calling the following functions.

- `feedback_prepare`
- `cgtfb_prepare`
- `compressor_prepare`

Subsequent signal processing is performed by calling the following functions.

- `cgtfb_analyze`
- `compressor_process`
- `cgtfb_synthesize`
- `feedback_record`

Several examples test basic aspects of these functions.

- `tst_gfa` – tests gammatone filter-bank analysis
- `tst_gfio` – tests simple waveform processing
- `tst_gfsc` – tests simple waveform processing with soundcard

To simulate gammatone filter-bank frequency analysis with instantaneous compression, variable initialization and memory allocation is performed by calling the following functions.

- `feedback_prepare`
- `firfb_prepare`
- `agc_prepare`

Subsequent signal processing is performed by calling the following functions.

- `agc_input`
- `firfb_analyze`
- `agc_channel`
- `firfb_synthesize`
- `agc_output`
- `feedback_record`

Several examples test basic aspects of these functions.

- `tst_ffa` – tests filter-bank analysis
- `tst_ffio` – tests simple waveform processing
- `gha_demo` – tests speech-waveform processing

All examples require the SIGPRO library from BTNRH (http://audres.org/rc/sigpro). The soundcard examples also require the ARSC library (http://audres.org/rc/arsc).

CHAPRO FUNCTION DESCRIPTIONS

cha_allocate

Allocates memory attached to CHAPRO data structure.

(void) **cha_allocate**(CHA_PTR **cp**, int **cnt**, int **siz**, int **idx**)

Function arguments

cp	pointer to CHAPRO data structure
cnt	Number of elements to allocate.
siz	Size of each element.
idx	Index into CHAPPRO data structure.

Return Value

none

Remarks

A pointer to the allocated memory is stored in the CHAPRO data structure at the location specified by **idx**.

See Also

cha_cleanup

cha_cleanup

Frees all memory attached to CHAPRO data structure.

(void) **cha_cleanup**(CHA_PTR **cp**)

Function arguments

cp	pointer to CHAPRO data structure

Return Value

none

Remarks

Should always be the last function called in the CHAPRO library.

See Also

cha_allocate

file specified by **fn**.

cha_data_gen

Generates C code that represents the CHAPRO data structure.

(int) **cha_data_gen**(CHA_PTR **cp**, char ***fn**)

Function arguments

cp	pointer to CHAPRO data structure
fn	Pointer to output filename.

Return Value

Error code:
0 – no error
1 – can’t open output file
2 – data structure not yet initialized
3 – data structure contains no data

Remarks

The C code generated by this function represents the CHAPRO data structure after variables have been initialized and data memory has been allocated by prior calls to any preparation functions. The code is written to the to the file specified by **fn**.

cha_fft_cr

Inverse Fourier transform complex frequency components into real signal.

(int) **cha_fft_cr**(float ***x,** int **n**)

Function arguments

x Complex frequency components are replaced by real-valued signal.

n Number of points in the signal.

Return Value

Error code:
0 – no error
1 **–** **n** not a power of 2

Remarks

The input array must be dimensioned to accommodate **n+2** float values. The number of complex frequency components is **(n+2)/2.**

cha_fft_rc

Fourier transform real signal into complex frequency components.

(int) **cha_fft_rc**(float ***x,** int **n**)

Function arguments

x	Real-valued signal is replaced by complex frequency components
n	Number of points in the signal.

Return Value

Error code:
0 – no error
1 **–** **n** not a power of 2

Remarks

The input array must be dimensioned to accommodate **n+2** float values. The number of complex frequency components is **(n+2)/2.**

cha_prepare

CHAPRO data structure preparation function.

(void) **cha_prepare**(CHA_PTR **cp**)

Function arguments

cp	pointer to CHAPRO data structure

Return Value

None.

Remarks

Should be called only once and prior to calling other library functions; however, violations of this rule may be tolerated.

cha_version

Returns a string that describes the current version of the CHAPRO library.

(char *) **cha_version**(void)

Function arguments

none

Return Value

Pointer to version string.

Remarks

An example of the return value, "CHAPro version 0.03, 6-Nov-2016".

cha_agc_prepare

Automatic-gain-control preparation function.

(int) **cha_agc_prepare**(CHA_PTR **cp**, CHA_DSL ***dsl**, CHA_WDRC ***gha**,
double **scale**)

Function arguments

cp	pointer to CHA data structure
dsl	pointer to DSL prescription structure (see Appendix C)
gha	pointer to WDRC prescription structure (see Appendix D)
scale	scale factor applied to input signal

Return Value

Error code:
0 – no error

Remarks

Initializes variables and allocates memory for automatic gain control. Chunk size is the number of samples read from the input signal and written to the output signal with each call to cha_agc_process.

See Also

cha_agc_process

cha_agc_input

Automatic-gain-control processing function.

(void) **cha_agc_input**(CHA_PTR **cp**, float ***x**, float ***y**, int **cs**)

Function arguments

cp	pointer to CHA data structure
x	pointer to input signal
y	pointer to output signal
cs	chunk size

Return Value

none

Remarks

Performs single-channel, automatic-gain-control processing with scale factor applied to input signal. Chunk size is the number of samples read from the input signal and written to the output signal.

See Also

cha_agc_prepare

cha_agc_channel

Automatic-gain-control processing function.

(void) **cha_agc_channel**(CHA_PTR **cp**, float ***x**, float ***y**, int **cs**)

Function arguments

cp	pointer to CHA data structure
x	pointer to input signal
y	pointer to output signal
cs	chunk size

Return Value

none

Remarks

Performs multi-channel, automatic-gain-control processing. Chunk size is the number of samples read from the input signal and written to the output signal.

See Also

cha_agc_prepare

cha_agc_output

Automatic-gain-control processing function.

(void) **cha_agc_output**(CHA_PTR **cp**, float ***x**, float ***y**, int **cs**)

Function arguments

cp	pointer to CHA data structure
x	pointer to input signal
y	pointer to output signal
cs	chunk size

Return Value

none

Remarks

Performs single-channel, automatic-gain-control processing (with no scale factor applied to input signal). Chunk size is the number of samples read from the input signal and written to the output signal.

See Also

cha_agc_prepare

cha_compressor_prepare

Instantaneous-compression preparation function.

(int) **cha_compressor_prepare**(CHA_PTR **cp**, float ***Lc**, float ***Gc**,
double **lr**, int **np**, int **ds**)

Function arguments

cp	pointer to CHA data structure
Lc	pointer to level array
Lc	pointer to gain array
lr	level reference
np	number of points in level and gain arrays
ds	down-sample factor

Return Value

Error code:
0 – no error

Remarks

Initializes variables and allocates memory for instantaneous compression. Chunk size is the number of samples read from the input signal and written to the output signal with each call to cha_compressor_process.

See Also

cha_compressor_process

cha_compressor_process

Instantaneous-compression processing function.

(void) **cha_compressor_process**(CHA_PTR **cp**, float ***x**, float ***y**, int **cs**)

Function arguments

cp	pointer to CHA data structure
x	pointer to input signal
y	pointer to output signal
cs	chunk size

Return Value

none

Remarks

Performs automatic-gain-control processing. Chunk size is the number of samples read from the input signal and written to the output signal.

See Also

cha_compressor_prepare

cha_feedback_prepare

Save output signal for feedback management.

(int) **cha_feedback_prepare**(CHA_PTR **cp**, int **cs**)

Function arguments

cp	pointer to CHAPRO data structure
cs	chunk size

Return Value

Error code:
0 – no error

Remarks

This function has not yet been implemented! The current list of function arguments is incomplete. Initializes variables and allocates memory for automatic gain control. Chunk size is the number of input samples that will be available to **cha_feedback_manage** and the number of output samples that will be available to **cha_feedback_record**.

See Also

cha_feedback_manage, cha_feedback_record

cha_feedback_manage

Process input signal to remove feedback.

(void) **cha_feedback_manage**(CHA_PTR **cp**, float ***x**, float ***y**, int **cs**)

Function arguments

cp	pointer to CHAPRO data structure
x	pointer to input signal
y	pointer to output signal
cs	chunk size

Return Value

none

Remarks

This function has not yet been implemented! Performs feedback management. Chunk size is the number of samples read from the input signal and written to the output signal.

See Also

cha_feedback_prepare, cha_feedback_record

cha_feedback_record

Save output signal for feedback management.

(void) **cha_feedback_record**(CHA_PTR **cp**, float ***x**, int **cs**)

Function arguments

cp	pointer to CHAPRO data structure
x	pointer to input signal
cs	chunk size

Return Value

none

Remarks

This function has not yet been implemented! Assists feedback management. Chunk size is the number of samples read from the input signal and written to the output signal.

See Also

cha_feedback_prepare, cha_feedback_manage

cha_cgtfb_prepare

Gammatone filter-bank preparation function.

(int) **cha_cgtfb_prepare**(CHA_PTR **cp**, double *__fc__, double *__bw__,
double **sr**, double **gd**, double **tw**, int **nc**, int **cs**, int **nmic**, int **nrec**)

Function arguments

cp	pointer to CHA data structure
fc	pointer to list of center frequencies (Hz)
bw	pointer to list of bandwidths (Hz)
sr	sampling rate (samples/second)
gd	target group delay (ms)
tw	buffer length (ms) for determining zero gain
nc	number of channels
cs	chunk size
nmic	number of microphones
nrec	number of receivers

Return Value

Error code:
0 – no error

Remarks

Initializes variables and allocates memory for the gammatone filter-bank. Chunk size is the number of samples read from the input signal and written to the output signal with each call to cha_compressor_process.

See Also

cha_cgtfb_analyze, cha_cgtfb_synthesize

cha_cgtfb_analyze

Gammatone filter-bank frequency-analysis function.

(void) **cha_cgtfb_analyze**(CHA_PTR **cp**, float ***x**, float ***y**, int **cs**)

Function arguments

cp	pointer to CHA data structure
x	pointer to input signal
y	pointer to output signal
cs	chunk size

Return Value

none

Remarks

Performs automatic-gain-control processing. Chunk size is the number of samples read from the input signal and written to the output signal.

See Also

cha_cgtfb_prepare, cha_cgtfb_synthesize

cha_cgtfb_synthesize

Gammatone filter-bank frequency-synthesis function.

(void) **cha_cgtfb_synthesize**(CHA_PTR **cp**, float ***x**, float ***y**, int **cs**)

Function arguments

cp	pointer to CHA data structure
x	pointer to input signal
y	pointer to output signal
cs	chunk size

Return Value

none

Remarks

Performs automatic-gain-control processing. Chunk size is the number of samples read from the input signal and written to the output signal.

See Also

cha_cgtfb_prepare, cha_cgtfb_analyze

cha_firfb_prepare

FIR filter-bank preparation function.

(int) **cha_firfb_prepare**(CHA_PTR **cp**, double ***cf**, int **nc**, double **sr**,
int **nw**, int **wt**, int **cs,** int **nmic,** int **nrec**)

Function arguments

cp	pointer to CHA data structure
cf	list frequency band edges (kHz)
nc	number of frequency bands
sr	sampling rate (samples/second)
nw	window size (samples)
wt	window type (0=Hamming, 1=Blackman)
cs	chunk size
nmic	number of microphones
nrec	number of receivers

Return Value

Error code:
0 – no error

Remarks

Initializes variables and allocates memory for the FIR filter-bank. Chunk size is the number of samples read from the input signal and written to the output signal with each call to cha_compressor_process.

See Also

cha_firfb_analyze, cha_firfb_synthesize

cha_firfb_analyze

FIR filter-bank frequency-analysis function.

(void) **cha_firfb_analyze**(CHA_PTR **cp**, float ***x**, float ***y**, int **cs**)

Function arguments

cp	pointer to CHA data structure
x	pointer to input signal
y	pointer to output signal
cs	chunk size

Return Value

none

Remarks

Performs FIR filter-bank analysis. Chunk size is the number of samples read from the input signal and written to the output signal.

See Also

cha_firfb_prepare, cha_firfb_synthesize

cha_firfb_synthesize

FIR filter-bank frequency-synthesis function.

(void) **cha_firfb_synthesize**(CHA_PTR **cp**, float ***x**, float ***y**, int **cs**)

Function arguments

cp	pointer to CHA data structure
x	pointer to input signal
y	pointer to output signal
cs	chunk size

Return Value

none

Remarks

Performs FIR filter-bank synthesis. Chunk size is the number of samples read from the input signal and written to the output signal.

See Also

cha_firfb_prepare, cha_firfb_analyze

Appendix A. Test programs

Several examples that test basic aspects of complex-gammatone filter-bank and instantaneous-compression functions.

- `tst_gfa` – tests complex-gammatone filter-bank analysis
- `tst_gfio` – tests simple waveform processing
- `tst_gfsc` – tests simple waveform processing with soundcard

Several examples that test basic aspects of FIR filter-bank and automatic-gain-control functions.

- `tst_ffa` – tests FIR filter-bank analysis
- `tst_ffio` – tests simple waveform processing
- `gha_demo` – tests speech-waveform processing

Appendix B. CLS Prescription

Structure CHA_CLS specifies the CLS prescription.

```
#define CLS_MXCH 32         // maximum number of channels

typedef struct {
    int cm;                 // compression mode
    int nc;                 // number of channels
    double fc[CLS_MXCH];    // center frequency
    double bw[CLS_MXCH];    // bandwith
    double Gcs[CLS_MXCH];   // gain at compression start
    double Gcm[CLS_MXCH];   // gain at compression middle
    double Gce[CLS_MXCH];   // gain at compression end
    double Gmx[CLS_MXCH];   // maximum gain
    double Lcs[CLS_MXCH];   // level at compression start
    double Lcm[CLS_MXCH];   // level at compression middle
    double Lce[CLS_MXCH];   // level at compression end
    double Lmx[CLS_MXCH];   // maximum output level
} CHA_CLS;
```

Appendix C. DSL Prescription

Structure CHA_DSL specifies the DSL prescription.

```
#define DSL_MXCH 32              // maximum number of channels

typedef struct {
    double attack;               // attack time (ms)
    double release;              // release time (ms)
    double maxdB;                // maximum output (dB SPL)
    int ear;                     // 0=left, 1=right
    int nchannel;                // number of channels
    double cross_freq[DSL_MXCH]; // cross frequencies (Hz)
    double tkgain[DSL_MXCH];     // compression-start gain
    double cr[DSL_MXCH];         // compression ratio
    double tk[DSL_MXCH];         // compression-start kneepoint
    double bolt[DSL_MXCH];       // broadband output limiting threshold
} CHA_DSL;
```

Appendix D. WDRC Parameters

Structure CHA_WDRC specifies single-channel WDRC parameters

```
typedef struct {
    double attack;               // attack time (ms)
    double release;              // release time (ms)
    double fs;                   // sampling rate (Hz)
    double maxdB;                // maximum signal (dB SPL)
    double tkgain;               // compression-start gain
    double tk;                   // compression-start kneepoint
    double cr;                   // compression ratio
    double bolt;                 // broadband output limiting threshold
} CHA_WDRC;
```