Ambiente de Teste para Filtros Adaptativos

Generated by Doxygen 1.7.6.1

Tue Dec 17 2013 05:58:29

Contents

Chapter 1

Todo List

Member main (int argc, char ∗argv[])

get input files from command-line.

Member portaudio_init ()

make the device listing optional

Class Signal

‘float‘s should be a typedef sample_t (since we don’t want a template)

Implement "stream" signals, to provide real-time processing.

File Signal.h

Separate implementation and declarations in different files.

Member Signal::copyfrom (Signal &other)

This method should be a C++ copy-constructor. Also, make sure ‘other‘ is ‘_const_ Signal&‘.

Member Signal::data

All uses of ‘data‘ are already encapsulated inside the Signal class implementation. This should be private.

Consider making this a std::vector, or std::valarray.

Member Signal::filter (Signal &imp_resp, Signal &conv)

Resolve possible sample_rate conflicts before filtering, using the same approach as in ‘Signal::add()‘

‘imp_resp‘ should be ‘_const_ Signal&‘.

Implement a DFT method, and rewrite this using overlap-and-save or overlap-and-add.

Find a way of returning ‘conv‘ without it getting destroyed at stack unwinding.

Member Signal::sample_rate

Can we make this a private member?

Member Signal::samples

Encapsulate (if they’re not already) all uses of ‘samples‘ inside the Signal class implementation. Then make this private.

Member Signal::Signal (std::string filename)

Should test ‘buf‘ for ‘malloc‘ error.

Should throw a more catchable exception at file open failure.

Chapter 2

Class Index

2.1 Class List

Here are the classes, structs, unions and interfaces with brief descriptions:

Chapter 3

File Index

3.1 File List

Here is a list of all documented files with brief descriptions:

Chapter 4

Class Documentation

4.1 Signal Class Reference

A time- or frequency-domain signal.

`#include <Signal.h>`

Public Types

- enum delay_type { MS, SAMPLE }

Public Member Functions

- Signal ()

  Constructs an empty signal.
- Signal (std::string filename)

  Constructs a signal from an audio file.
- ∼Signal ()

  Frees memory used.
- void copyfrom (Signal &other)

  Constructs a signal as a copy of another.
- float & operator[] (unsigned long index)

  Returns a sample.
- void set_size (unsigned long n)

  Changes the number of samples.
- void set_samplerate (int sr)

  Changes the signal sample rate.
- void delay (delay_type t, unsigned long d)

  Delays the signal in time.
- void add (Signal &other)

Adds the 'other' signal to the caller.

- void gain (float g)

  Applies gain 'g' to the signal.
- void filter (Signal &imp_resp, Signal &conv)

  Convolves the sinal.
- void play (bool sleep=true)

  Makes PortAudio playback the audio signal.

Public Attributes

- float ∗ data

  Pointer to the array of samples.
- unsigned long samples

  Number of samples.
- int sample_rate

  Signal sample rate in hertz
- unsigned long counter

  general-purpose variable for external use.

4.1.1 Detailed Description

A time- or frequency-domain signal.

Holds data and provides routines for dealing with time-domain and frequency-domain signals. Currently, all Signals are an array of single-precision floating-point samples. Signals know their sample rate.

Todo 'float's should be a typedef sample_t (since we don't want a template)

Todo Implement "stream" signals, to provide real-time processing.

Definition at line 41 of file Signal.h.

4.1.2 Member Enumeration Documentation

4.1.2.1 enum Signal::delay_type

This is a type for specifying whether a time interval is given in milliseconds or in samples.

Enumerator:

MS Time interval given in milliseconds.

SAMPLE Time interval given in samples.

Definition at line 48 of file Signal.h.

4.1.3 Constructor & Destructor Documentation

4.1.3.1 Signal::Signal ( )

Constructs an empty signal.

Initializes the signal with no meta-data and no samples. The user needs to specify the sample rate and create samples before using the signal.

Definition at line 107 of file Signal.h.

4.1.3.2 Signal::Signal (std::string *filename*)

Constructs a signal from an audio file.

Constructs a signal getting the signal data from an audio file. This is done using the [libsndfile][libsndfile] library. The filetypes supported are listed [here][libsndfile_-features]. WAV is supported, but MP3 is not.

If the given file is stereo, of multi-channel, just the first channel will be read. (On stereo audio files, this is the left channel.)

The sample rate is extracted from the file's meta-data info.

Parameters

`in`	*filename*	Audio file name.

Exceptions

'std::runtime_error'	if file openening fails.
'std::runtime_error'	if file reading fails.

Todo Should test 'buf' for 'malloc' error.

Todo Should throw a more catchable exception at file open failure.

Definition at line 201 of file Signal.h.

References data, sample_rate, samples, and set_size().

4.1.3.3 Signal::~Signal ( )

Frees memory used.

If the signal is not empty, free the pointer to the array of samples.

Definition at line 115 of file Signal.h.

References data.

4.1.4 Member Function Documentation

4.1.4.1 void **Signal::add** (**Signal** & *other*)

Adds the ‘other‘ signal to the caller.

Adds the ‘other‘ signal to the caller signal. First, we re-sample ‘other‘ into a new temporary signal. Then we increase the caller’s size if needed, and finally add the signals sample-by-sample.

Parameters

`in`	*other*	The signal to be added to the caller.

Definition at line 453 of file Signal.h.

References copyfrom(), sample_rate, samples, set_samplerate(), and set_size().

Referenced by main().

4.1.4.2 void **Signal::copyfrom** (**Signal** & *other*)

Constructs a signal as a copy of another.

Constructs a signal as a copy of another one. If this signal is not empty, we destroy it.

Parameters

`in`	*other*	The signal to be copied from.

Exceptions

‘std::runtime_error‘	if memory allocation fails. Memory allocation happens because the signal sizes might differ.

Todo This method should be a C++ copy-constructor. Also, make sure ‘other‘ is ‘_-const_ Signal&‘.

Definition at line 169 of file Signal.h.

References data, sample_rate, and samples.

Referenced by add().

4.1.4.3 void **Signal::delay** (**delay_type** *t*, unsigned long *d*)

Delays the signal in time.

Adds zeroed samples at the beginning of the signal.

Parameters

`in`	*t*	A ‘delay_type‘ element.
`in`	*d*	The time interval to be delayed, given in the units specified by ‘t‘.

Exceptions

‘std::runtime_error‘	if memory realloc fails.

Definition at line 245 of file Signal.h.

References data, MS, sample_rate, samples, and set_size().

Referenced by main().

4.1.4.4 void Signal::filter (Signal & *imp_resp,* Signal & *conv*)

Convolves the sinal.

Generates a new signal, which is the convolution of the caller signal and a given filter impulse response (FIR).

Parameters

`in`	*imp_resp*	The filter impulse response to be convolved with.
`out`	*conv*	The resulting signal.

Exceptions

‘std::runtime_error‘	if memory alloc fails.

Todo Resolve possible sample_rate conflicts before filtering, using the same approach as in ‘Signal::add()‘

Todo ‘imp_resp‘ should be ‘_const_ Signal&‘.

Todo Implement a DFT method, and rewrite this using overlap-and-save or overlap-and-add.

Todo Find a way of returning ‘conv‘ without it getting destroyed at stack unwinding.

Definition at line 282 of file Signal.h.

References data, sample_rate, samples, and set_size().

Referenced by main().

4.1.4.5 void Signal::gain (float *g*)

Applies gain ‘g‘ to the signal.

Apply a gain ‘g‘ to the signal. This can be useful, for example, to make sure that the signal is in the [-1, 1] range.

Parameters

in	*g*	The signal gain to be applied.

Definition at line 469 of file Signal.h.

References samples.

Referenced by main().

4.1.4.6 float & Signal::operator[] (unsigned long *index*) `[inline]`

Returns a sample.

Gets a sample of the signal. For performance reasons, this method does not check that the given index is valid.

Parameters

in	*index*	The index of the desired sample. Signal indexes are zero-based.

Returns

a reference to the sample.

Definition at line 130 of file Signal.h.

References data.

4.1.4.7 void Signal::play (bool *sleep* = `true`)

Makes PortAudio playback the audio signal.

Creates a PortAudio stream for audio playback of the signal content. If ‘sleep‘ is ‘true‘, we wait for the playback to end before returning. (If it’s false, the function returns, while playback goes on in the background.)

Parameters

in	*sleep*	Whether or not to sleep before returning.

Exceptions

std::runtime_error	if any of the PortAudio steps fail (check the source code)

See also

callback

Definition at line 373 of file Signal.h.

References sample_rate, and samples.

Referenced by main().

4.1.4.8 void **Signal::set_samplerate** (int *sr*)

Changes the signal sample rate.

Changes the sample rate of the signal. First, we reconstruct the time-domain signal by linear interpolation. Then, we re-sample the continuous-time reconstructed signal at the new sample rate.

Parameters

`in`	*sr*	The new sample rate in Hertz.

Exceptions

std::runtime_error	if memory alloc fails

See also

Signal::sample_rate

Definition at line 430 of file Signal.h.

References sample_rate, and samples.

Referenced by add().

4.1.4.9 void **Signal::set_size** (unsigned long *n*)

Changes the number of samples.

Changes the signal length. Allocates more space if we are growing the signal, and deletes the last samples if we are shrinking it.

Parameters

`in`	*n*	The desired signal length.

Exceptions

‘std::runtime_error‘	if the memory reallocation fails.

Definition at line 147 of file Signal.h.

References data, and samples.

Referenced by add(), delay(), filter(), main(), and Signal().

4.1.5 Member Data Documentation

4.1.5.1 float* Signal::data

Pointer to the array of samples.

This member is public only because we need to pass it to the libsndfile read audio file function.

Todo All uses of ‘data‘ are already encapsulated inside the Signal class implementation. This should be private.

Todo Consider making this a std::vector, or std::valarray.

Definition at line 68 of file Signal.h.

Referenced by copyfrom(), delay(), filter(), operator[](), set_size(), Signal(), and ∼Signal().

4.1.5.2 int Signal::sample_rate

Signal sample rate in hertz

Todo Can we make this a private member?

Definition at line 84 of file Signal.h.

Referenced by add(), copyfrom(), delay(), filter(), play(), set_samplerate(), and Signal().

4.1.5.3 unsigned long Signal::samples

Number of samples.

This member is public only because we need to pass it to libsndfile functions.

Todo Encapsulate (if they’re not already) all uses of ‘samples‘ inside the Signal class implementation. Then make this private.

Definition at line 78 of file Signal.h.

Referenced by add(), copyfrom(), delay(), filter(), gain(), play(), set_samplerate(), set_size(), and Signal().

The documentation for this class was generated from the following file:

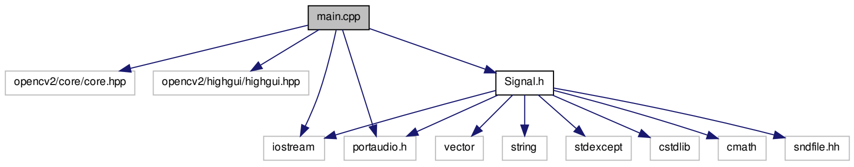
- Signal.h (aaa.versao.teste.1-7-g6a5c853)

Chapter 5

File Documentation

5.1 main.cpp File Reference

`#include <opencv2/core/core.hpp>` `#include <opencv2/highgui/highgui.-hpp>` `#include <iostream>` `#include <portaudio.h>` `#include "Signal.h"` Include dependency graph for main.cpp:



Functions

- void portaudio_init ()

  Initialize PortAudio.
- void portaudio_end ()

  Close PortAudio.
- int main (int argc, char ∗argv[ ])

  'main()' function.

5.1.1 Detailed Description

Holds the 'main()' function and other routines.

Author

Pedro Angelo Medeiros Fonini

Definition in file main.cpp.

5.1.2 Function Documentation

5.1.2.1 int main (int *argc*, char ∗ *argv[]*)

‘main()‘ function.

No command-line parameters yet.

This function: 1. Prints version info 2. Creates two ‘Signal‘s, ‘sound_me‘ and ‘sound_other‘ from the two input files. 3. Delays the second. 4. Creates an impulse response. 5. Creates a new Signal ‘signal_result‘ which is the first filtered, added to the second, delayed. 6. Initializes a PortAudio session, plays the resulting sound, and closes PortAudio.

Todo get input files from command-line.

Parameters

`in`	*argc*	argument count (unused)
`in`	*argv*	argument values (unused)

Returns

0 if no errors

Definition at line 111 of file main.cpp.

References Signal::add(), Signal::delay(), Signal::filter(), Signal::gain(), Signal::MS, -Signal::play(), portaudio_end(), portaudio_init(), and Signal::set_size().

5.1.2.2 void portaudio_end ( )

Close PortAudio.

Ends a PortAudio session.

Exceptions

‘std::runtime_error‘	if PortAudio closing fails.

See also

'portaudio_init()'

Definition at line 81 of file main.cpp.

Referenced by main().

5.1.2.3 void portaudio_init ( )

Initialize PortAudio.

Initializes a PortAudio session. Also prints out a list of available devices that PortAudio sees.

Exceptions

'std::runtime_error'	if PortAudio initialization fails.

See also

'portaudio_end()'

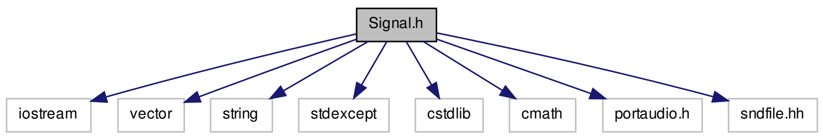
Todo make the device listing optional

Definition at line 41 of file main.cpp.

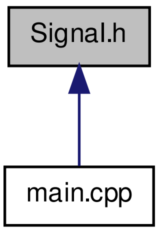
Referenced by main().

5.2 Signal.h File Reference

`#include <iostream> #include <vector> #include <string> #include <stdexcept> #include <cstdlib> #include <cmath> × #include <portaudio.h> #include <sndfile.hh>` Include dependency graph for Signal.h:



This graph shows which files directly or indirectly include this file:



Classes

- class Signal

  A time- or frequency-domain signal.

5.2.1 Detailed Description

Holds everything to do with the ‘Signal‘ class.

Todo Separate implementation and declarations in different files.

Author

Pedro Angelo Medeiros Fonini

Definition in file Signal.h.