STEMS-PRO 1.0.0

by zplane.development

December 6, 2023

Contents

1 STEMS PRO Documentation

1.1 Introduction

STEMS PRO is a source separation and stem generation algorithm that is capable of separating any stereo or mono mixture into vocals, bass, drums, and ”other” (4 stereo or mono stems in total).

The underlying algorithm in STEMS PRO is based on a single deep neural network designed to deliver the highest quality while remaining fast enough for most applications. A *standard* mode and a *high_quality* mode are available, described in the section 1.2.1. Internally, STEMS PRO works by analyzing small chunks of audio one at a time and estimating all instruments for that audio chunk. The STEMS PRO interface allows the user to set the output block size according to their needs. Additionally, enhancement can be applied to each instrument; this will activate a dedicated ”enhancer” deep neural network, for the selected instrument, which was specifically designed to reduce artifacts. Due to the nature of the underlying neural network, it can not be guarantied that the sum of outputs will produce exactly the initial mixture. For this reason STEMS PRO has the option to output the *residual*, which is simply the difference between the stems and the initial mixture, on extra channels. When the *residual* is mixed with the instrument outputs of STEMS PRO, the result yields exactly the initial mixture , which can be useful in certain cases. An overview of the algorithm is shown below.

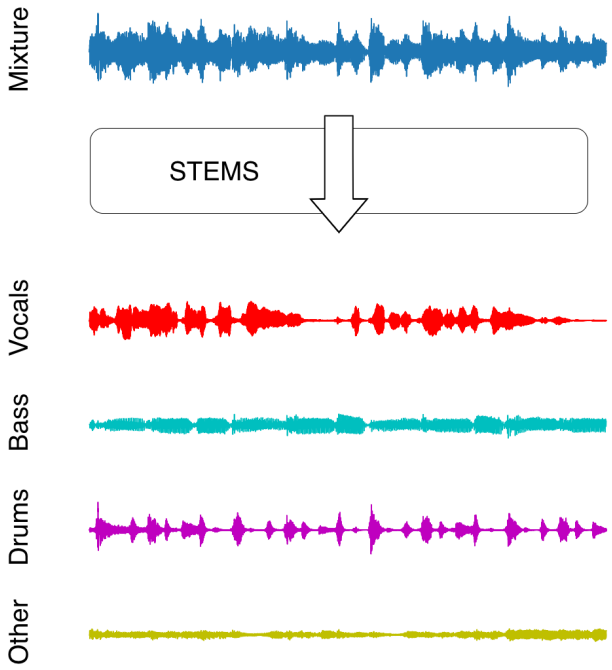


Figure 1: Overview of the source separation algorithm

This document contains all information you need to get started and use STEMS PRO to the best of its abilities. At its core are in-depth descriptions of the API through usage examples provided in the StemsProClMain.cpp. This example demonstrate all

the functionalities of STEMS PRO and can be used to get started as quickly and easily as possible. What follows is a detailed documentation of the STEMS PRO interface along with all the methods and structs contained within.

1.2 API Documentation

1.2.1 Stems Modes

The STEMS PRO API has two modes: *standard* and *high_quality* defined in **enum StemsPro::Mode**. We recommend to use the *standard* mode for most applications where processing time is critical. The *high_quality* mode provides the best quality at a loss of processing speed. Both modes should perform faster than real-time on most modern hardware.

1.2.2 Memory Allocation

The STEMS PRO SDK does not allocate buffers handled by the calling application; the input buffers must be allocated by the calling application. Input audio buffers are allocated as double arrays of size [numChannels][maxBlockSize] where maxBlockSize should be the value returned by **StemsPro::getMaxFramesNeeded()**. Note that the value returned by **StemsPro::getMaxFramesNeeded()** depends on the input sampling rate. The 4 or 5 output buffers (one for each instrument plus optional *residual*), are allocated as double arrays of size [numChannels][blockSize] where blockSize is the output block size chosen by the user.

1.2.3 Naming Conventions

When talking about **frames**, the number of audio samples per channel is meant. For example, 512 stereo frames correspond to 1024 float values (samples). If the sample size is 32bit float, one sample has a memory usage of 4 bytes. An audio **block** is a sequence of consecutive frames with a given length.

1.2.4 Instance Handling Functions

- **ErrorType StemsPro::initialize (int numChannels, float sampleRate, int max↩ Outputblocksize, int numThreads, StemsPro::Mode mode, bool residual);**
  - Initalizes a STEMS PRO instance.
- **ErrorType StemsPro::reset() ;**
  - Resets all internal variables and buffers to the default state. The return value indicates whether an error occurred or not.

1.2.5 Process Functions

- **ErrorType StemsPro::process(float const∗ const∗ const ppfInputBlock, std↩ ::size_t numInputFrames, float∗ const∗ const ppfOutputBlock)**
  - Performs the actual stems separation processing if the number of frames provided is as retrieved by **StemsPro::getFramesNeeded()**.

- **ErrorType StemsPro::finishProcessing(float const* const* const ppfInput↩ Block, std::size_t numInputFrames)**
  - Signals the end of the processing loop. Input contains the remaining samples of the input signal that should be processed. numInputFrames must be less than the length reported by **StemsPro::getFramesNeeded()**, otherwise process can still be called. This function must only be called once. After a call to this function, no further calls to process() are possible.
- **ErrorType StemsPro::flushBuffer (float* const* const ppfOutputBlock, std↩ ::size_t& numOutputFrames)**
  - Gets all the remaining internal frames when no more input data is available and writes them into the buffer ppfOutputBlock. Returns the number of written samples. The use of this function is optional.

1.2.6 Parameter Retrieving and Setting Functions

- **ErrorType StemsPro::setOutputBlockSize (std::size_t outputBlockSize)**
  - Sets the output block size.
- **std::size_t StemsPro::getFramesNeeded();**
  - Returns the required number of sample frames in order to obtain a full output block during the next call to StemsPro::process()
- **std::size_t StemsPro::getMaxFramesNeeded();**
  - Returns the maximum required number of frames needed.
- **void StemsPro::setEnableEnhancementOnInstrument (Instrument instrument, bool activateEnhancement);**
  - Applies enhancement on a selected instrument. The enhancement here improves separation and perceptual quality for the selected instrument. Enhancement can be activated or deactivated during processing. Enhancement can also be activated for multiple instruments.
- **bool StemsPro::getEnableEnhancementOnInstrument (Instrument instrument);**
  - Returns information on the current activation state of enhancement for a given instrument.

1.2.7 C++ Usage example demonstrating the functionalities of a single instance

The complete code referenced here can be found in the example source file StemsProClMain.cpp. An overview of the processing steps and buffer management is illustrated below.

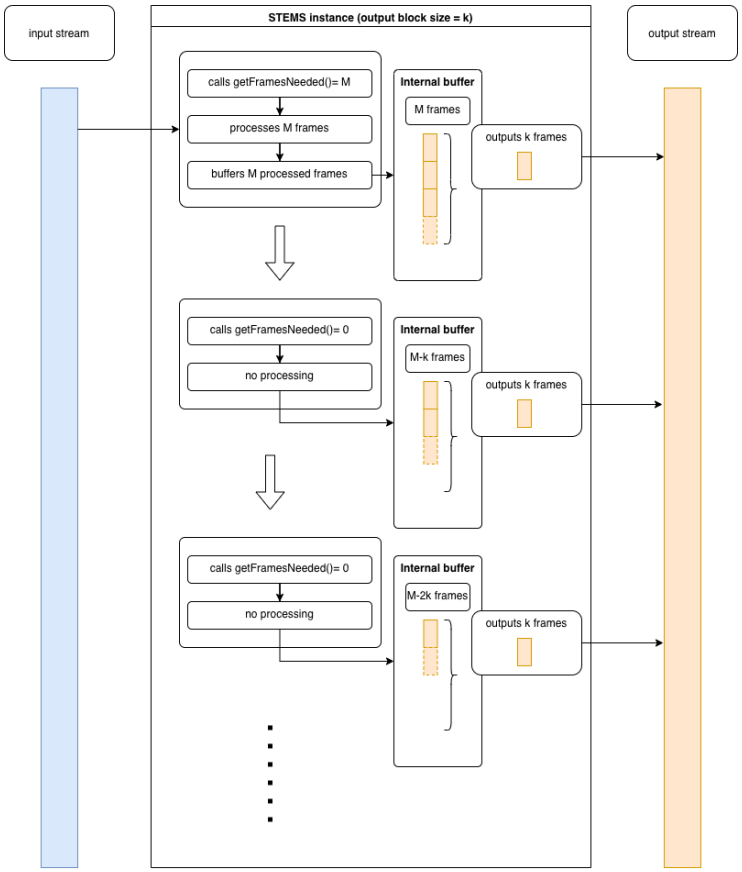


Figure 2: Flow chart illustration

In the first step, an instance of `STEMS PRO` must be created. **Note that this is different to how you would create an instance of previous zplane SDKs**. After instantiation, we use the **StemsPro::initialize()** method to prepare `STEMS PRO` to be used.

```
zplane::StemsPro stems;
error = stems.initialize (inputFile.GetNumOfChannels(), inputFile.GetSampleRate(), kBlockSize
  , numThreads, mode, residual);
```

In this case, we initialize the instance of `STEMS PRO` with the channels and samplerate from the audio file to be processed, as well as the number of threads or cpu cores we want to use simultaneously for processing and the mode in which to operate. We also have to set the desired output block size here. After initializing, we can optionally set the enhancement to certain instruments. This can also be done later on.

```
stems.setEnableEnhancementOnInstrument (
  zplane::StemsPro::vocals, enhancementOnVocals);
```

Now that we have initialized our instance of `STEMS PRO` , we can allocate our input buffer according to the maximum of frames needed by the instance:

```
ppfInput[i] = new float[stems.getMaxFramesNeeded()];
```

The output buffers are allocated with the desired output block size, and with the total number of channels required by the STEMS PRO instance.

```
int numOutputChannels = static_cast<int> (stems.getNumOutputChannels());
```

And:

```
ppfOutput[i] = new float[kBlockSize];
```

Now that we have allocated all our buffers, we can start processing. We start the process loop by asking our STEMS PRO instance how many frames are needed for processing.

```
size_t inputSize = stems.getFramesNeeded();
```

We then read the audio file and fill the input buffer:

```
numFramesRead = inputFile.Read (ppfInput, inputSize);
```

The process function can then be called with the required number of input frames.

```
stems.process (ppfInput, numFramesRead, ppfOutput);
```

If there was no error, we can simply write out the output buffers for each instrument. Note that the output buffer has 8 channels for each of the 4 stereo stems, or 10 channels when residual output in enabled.

```
  vocalsFile.Write (&ppfOutput[stems.getInstrumentOutputChannels (
zplane::StemsPro::vocals)], kBlockSize);
  bassFile.Write (&ppfOutput[stems.getInstrumentOutputChannels (
zplane::StemsPro::bass)], kBlockSize);
  drumsFile.Write (&ppfOutput[stems.getInstrumentOutputChannels (
zplane::StemsPro::drums)], kBlockSize);
  otherFile.Write (&ppfOutput[stems.getInstrumentOutputChannels (
zplane::StemsPro::other)], kBlockSize);
  if (residual)
      residualFile.Write (&ppfOutput[stems.getInstrumentOutputChannels (
zplane::StemsPro::residual)], kBlockSize);
```

Once all available blocks have been fed to STEMS PRO and the last section of audio is smaller than number of frames required by the instance, the process loop is exited.

We can now call **StemsPro::finishProcessing()** to process the last section of audio.

```
stems.finishProcessing (ppfInput, numFramesRead);
```

Afterwards, to get the remaining samples in the internal buffer of STEMS PRO , we need to call the **StemsPro::flushBuffer()** method and write the samples to the output files.

```
stems.flushBuffer (ppfOutput, numOfOutputFrames);
```

After successful processing, we can simply close the audio files.

1.3 Support

Support for the source code is - within the limits of the agreement - available from:

zplane.development
Grunewaldstr. 83
d-10823 berlin
Germany

fon: +49.30.854 09 15.0
fax: +49.30.854 09 15.5

@: info@zplane.de

2 Namespace Index

2.1 Namespace List

Here is a list of all namespaces with brief descriptions:

3 Class Index

3.1 Class List

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

4 File Index

4.1 File List

Here is a list of all files with brief descriptions:

5 Namespace Documentation

5.1 zplane Namespace Reference

Classes

- class StemsPro

6 Class Documentation

6.1 zplane::StemsPro Class Reference

`#include <StemsPro/StemsPro.h>`

Public Types

- enum ErrorType {
  noError, memError, notInitializedError, alreadyInitializedError,
  invalidFunctionParamError, invalidFunctionCallError, unknownError, invalid↩
  Instrument,
  numErrorTypes }
- enum VersionType {
  major, minor, patch, revision,
  numVersionTypes }
- enum Mode { high_quality, standard }
- enum Instrument {
  vocals, bass, drums, other,
  residual }

Public Member Functions

- StemsPro ()
- ∼StemsPro ()
- ErrorType initialize (int numChannels, float sampleRate, int maxOutputblocksize, int numThreads, Mode mode, bool residual)
- bool isInitialized ()
- ErrorType setOutputBlockSize (std::size_t outputBlockSize)
- std::size_t getFramesNeeded ()
- std::size_t getMaxFramesNeeded ()
- std::size_t getInstrumentOutputChannels (Instrument instrument)
- std::size_t getNumOutputChannels ()
- ErrorType process (float const ∗const ∗const ppfInputBlock, std::size_t num↩ InputFrames, float ∗const ∗const ppfOutputBlock)
- ErrorType finishProcessing (float const ∗const ∗const ppfInputBlock, std::size↩ _t numInputFrames)
- ErrorType flushBuffer (float ∗const ∗const ppfOutputBlock, std::size_t &num↩ OutputFrames)
- ErrorType reset ()
- StemsPro::ErrorType setEnableEnhancementOnInstrument (Instrument instrument, bool activateEnhancement)
- bool getEnableEnhancementOnInstrument (Instrument instrument)

Static Public Member Functions

- static const char ∗ getVersion ()
- static const char ∗ getBuildDate ()

6.1.1 Detailed Description

Definition at line 40 of file StemsPro.h.

6.1.2 Member Enumeration Documentation

ErrorType `enum zplane::StemsPro::ErrorType`

Enumerator

noError	no error occurred
memError	memory allocation failed
notInitializedError	instance has not been initialized yet
alreadyInitializedError	instance has already been initialized
invalidFunctionParamError	one or more function parameters are not valid
invalidFunctionCallError	function call is not allowed
unknownError	unknown error occurred
invalidInstrument	selected instrument not valid for enhancement
numErrorTypes	

Definition at line 43 of file StemsPro.h.

```
44         {
45             noError,
46             memError,
47             notInitializedError,
48             alreadyInitializedError,
49             invalidFunctionParamError,
50             invalidFunctionCallError,
51             unknownError,
52             invalidInstrument,
53             numErrorTypes
54         };
```

Instrument `enum zplane::StemsPro::Instrument`

Enumerator

vocals	
bass	
drums	
other	
residual	

Definition at line 71 of file StemsPro.h.

```
72         {
73             vocals,
```

```
74              bass,
75              drums,
76              other,
77              residual
78          };
```

Mode `enum zplane::StemsPro::Mode`

Enumerator

high_quality	
standard	

Definition at line 65 of file StemsPro.h.

```
66          {
67              high_quality,
68              standard
69          };
```

VersionType `enum zplane::StemsPro::VersionType`

Enumerator

major	
minor	
patch	
revision	
numVersionTypes	

Definition at line 56 of file StemsPro.h.

```
57          {
58              major,
59              minor,
60              patch,
61              revision,
62              numVersionTypes
63          };
```

6.1.3 Constructor & Destructor Documentation

StemsPro() `zplane::StemsPro::StemsPro ( )`

~StemsPro() `zplane::StemsPro::~StemsPro ( )`

6.1.4 Member Function Documentation

finishProcessing() `ErrorType zplane::StemsPro::finishProcessing ( float const *const *const ppfInputBlock, std::size_t numInputFrames )`

Signals the end of the processing loop. Input contains the remaining samples of the input signal that should be processed. numInputFrames must be less than the length reported by StemsPro::getFramesNeeded(), otherwise process can still be called. This function must only be called once. After a call to this function no further calls to StemsPro::process() are possible.

Parameters

ppfInputBlock	: input sample buffer [channels][samples]
numInputFrames	: the number of input frames

Returns

StemsPro::ErrorType : Returns an error flag

flushBuffer() `ErrorType zplane::StemsPro::flushBuffer ( float *const *const ppfOutputBlock, std::size_t & numOutputFrames )`

Gets all the remaining internal frames when no more input data is available and writes them into the buffer ppfOutputBlock. Returns the number of written samples. The use of this function is optional.

Parameters

ppfOutputBlock	output sample buffer [channels][samples]
numOutputFrames	the number of output frames

Returns

StemsPro::ErrorType : Returns an error flag

getBuildDate() `static const char* zplane::StemsPro::getBuildDate ( ) [static]`

Returns the build date string.

getEnableEnhancementOnInstrument() `bool zplane::StemsPro::getEnableEnhancementOnInstrument ( Instrument instrument )`

Returns information on the current activation state of enhancement for a given instrument.

Parameters

instrument	StemsPro::Instrument, instrument to apply enhancement to

Returns

activateEnhancement: bool, whether or not enhancement is activated for selected track

getFramesNeeded() `std::size_t zplane::StemsPro::getFramesNeeded ( )`

Returns the required number of sample frames in order to obtain a full output block during the next call to StemsPro::process()

Returns

size_t : required number of sample frames

getInstrumentOutputChannels() `std::size_t zplane::StemsPro::getInstrumentOutputChannels (`
`Instrument instrument )`

Returns the output channel of the selected instrument, where StemsPro output a multichannel buffer with different instruments assigned to different channels.

Parameters

instrument	

Returns

channel assigned to selected instrument

getMaxFramesNeeded() `std::size_t zplane::StemsPro::getMaxFramesNeeded ( )`

Returns the maximum required number of frames needed. This value is dependent on the output block size.

Returns

size_t : required number of sample frames

getNumOutputChannels() `std::size_t zplane::StemsPro::getNumOutputChannels ( )`

Returns the total number of output channels. This depends on the number of input channels (mono or stereo) and the number of StemsPro to extract (stems2, stems4).

Returns

total number of output channels

getVersion() `static const char* zplane::StemsPro::getVersion ( ) [static]`

Returns major version, minor version, patch and build number of this StemsPro version.

initialize()

```
ErrorType zplane::StemsPro::initialize (
    int numChannels,
    float sampleRate,
    int maxOutputblocksize,
    int numThreads,
    Mode mode,
    bool residual )
```

Initialize an instance of StemsPro. Must be called before using any of StemsPro functionality.

Parameters

numChannels	Number of channels in the input signal.
sampleRate	Sample rate of the input signal in Hertz.
maxOutputblocksize	Blocksize of the output buffer in samples.
numThreads	the number of threads, or cpu cores to allocate for inference
mode	inference mode of the Stems instance, can be ”high_quality”, or ”standard”.
residual	whether or not to output the residual, which is the difference between the mixture and the individual stems.

Returns

StemsPro::ErrorType : Returns an error flag

isInitialized() `bool zplane::StemsPro::isInitialized ( )`

process()

```
ErrorType zplane::StemsPro::process (
    float const *const *const ppfInputBlock,
    std::size_t numInputFrames,
    float *const *const ppfOutputBlock )
```

Performs the actual stems separation processing if the number of frames provided is as retrieved by StemsPro::getFramesNeeded(). Input needs to be stereo. Output is multichannel stereo with the stems stacked in channels. With vocals, bass, drums and other, output has 8 channels.

Parameters

ppfInputBlock	: input sample buffer [channels][samples]
numInputFrames	: the number of input frames
ppfOutputBlock	: output sample buffer [channels][samples]

Returns

StemsPro::ErrorType : Returns an error flag

reset() `ErrorType zplane::StemsPro::reset ( )`

Clears the internal buffers. Call this method to avoid the remaining samples in the process buffer being audible when you e.g. stop playback and start it again at different time position. Other parameters are not reset.

Returns

StemsPro::ErrorType : Returns an error flag

setEnableEnhancementOnInstrument() `StemsPro::ErrorType zplane::StemsPro↩ ::setEnableEnhancementOnInstrument ( Instrument instrument, bool activateEnhancement )`

Applies enhancement on selected instrument. The enhancement here reduces interferences from other instruments and improves separation for the selected instrument. Enhancement can be activated or turned off during processing. Enhancement can also be activated for different instruments.

Parameters

instrument	StemsPro::Instrument, instrument to apply enhancement to
activateEnhancement	bool, turns the enhancement on or off for the selected instrument.

setOutputBlockSize() `ErrorType zplane::StemsPro::setOutputBlockSize ( std::size_t outputBlockSize )`

Sets the output block size.

Parameters

outputBlockSize	: the new output blocksize

Returns

StemsPro::ErrorType : Returns an error flag

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

- StemsPro/StemsPro.h

7 File Documentation

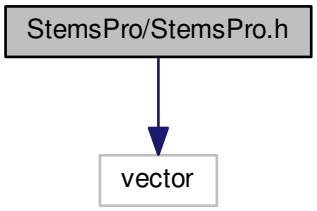
7.1 /work/project/docs/docugen.txt File Reference

7.2 StemsPro/StemsPro.h File Reference

interface of the StemsPro class.

```
#include <vector>
```

Include dependency graph for StemsPro.h:



Classes

- class zplane::StemsPro

Namespaces

- zplane

7.2.1 Detailed Description

interface of the StemsPro class.

:

Index