DRUMS 1.0.0

by zplane.development

April 22, 2025

Contents

1 DRUMS Documentation

1.1 Introduction

Welcome to zplane DRUMS, an SDK that analyzes audio recordings of drums and outputs a 9-class transcription of the signal, including velocities. DRUMS is an offline process, meaning it requires the complete audio file as input and outputs the result only after the file has been processed in its entirety. It is *not* an online or real-time drum transcription system. The algorithm is able to process audio files between 6 and 7 times realtime on an Apple M3 Pro processor. The mimimum sample rate is 11025 Hz and there is no restriction on the number of input channels. In order to maximize transcription quality, we offer the following guidelines when working with DRUMS:

- Isolated drums, i.e., no tonal instruments in the mix.
- Dry recording, with as little influence from room acoustics or FX processing as possible.
- Standard Western drum kit; DRUMS detects hits from each of the following classes: kick, snare, tom 1, tom 2, tom 3, open hi-hat, closed hi-hat, crash, and ride. Additional drum kit elements (such as cowbell) may be ignored or mis-classified, depending on their characteristics. While it is possible to analyze (electronic) drum machine recordings, this is currently out of scope and your results may vary.

This guide contains the information you need to get started using DRUMS: it provides an API description and C++ usage examples following a simple command-line application found in DrumsClMain.cpp. At the end, you will also find detailed documentation of the DRUMS∗ interface, describing all its methods and structs.

1.2 API Documentation

The analysis consists of two stages: a pre-processing stage in which the audio is analyzed, and a processing stage that carries out the actual drum transcription.
The pre-processing stage is based on the push principle: successive blocks of input audio frames are fed into the Drums::preProcess() function. Once the audio file has been loaded entirely, the pre-processing stage is ended by calling Drums::finishPreProcess(). After pre-processing is complete, Drums::process() is called in order to obtain the transcription results.

1.2.1 Memory Allocation

The DRUMS SDK does not allocate any buffers handled by the calling application. The input buffer as well as the result objects have to be allocated/created by the calling application.

1.2.2 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). An audio **block** is a sequence of consecutive frames with a given length.

1.2.3 Instance Handling Functions

- **ErrorType Drums::initialize (float sampleRate, std::size_t numChannels);**
  - Initalizes a DRUMS instance.
- **ErrorType Drums::reset() ;**
  - Resets all internal variables and buffers to the default state. The return value indicates whether an error occurred or not.

1.2.4 Process Functions

- **ErrorType preProcess (float const* const* const inputBuffer, int numberOfFrames)**
  - Pre-processing function. ppfInputBuffer is an array of pointers to the audio data. inputBuffer[0] is a pointer to the data of the first channel, ppfInputBuffer[1] points to the data of the second channel, etc. numberOfFrames specifies the number of frames, i.e., the number of samples in each channel. This function can repeatedly be called with successive chunks of audio until the entire signal has been pushed into DRUMS. This function will return an error if it is called after Drums::finishPreProcess() has been called.
- **ErrorType Drums::finishPreProcess (bool flushBuffers)**
  - This function has to be called after all audio frames have been pushed into DRUMS by Drums::preProcess(). It should be called exactly once and will return an error if called before Drums::preProcess() or after Drums::process(). Should you plan to add further audio to the buffer via preProcess(...), set the flushBuffers flag to false.
- **ErrorType Drums::process (std::vector<DrumResultElement>& result)**
  - Performs the actual drum transcription. This function can only be called after Drums::process() has been called. Returns a vector of drum kit element classes, indexed as shown in the table below. Note that in DrumsClMain.cpp we provide a default mapping and helper code to write MIDI files, but you are free to map each class to the MIDI note of your choice.

Table 1: Drum Kit Element Mapping Overview

Class Index	Drum Kit Element	Suggested MIDI Note
0	Kick	36
1	Snare	38
2	Tom 1	50
3	Tom 2	47
4	Tom 3	43
5	Open Hi-Hat	46
6	Closed Hi-Hat	42
7	Crash	49
8	Ride	51

1.2.5 Retrieving Parameters and Setting Functions

- **std::size_t Drums::getFramesNeeded();**
  - Returns the required number of frames in order to process a full block during the next call to Drums::process()
- **std::size_t Drums::getMaxFramesNeeded();**
  - Returns the maximum required number of frames.

1.3 Command Line Usage Example

The command line example can be executed by the following command

```
DrumsCl -i <inputFileWAV> -r <resultFileText> -m <resultFileMIDI>
```

The complete code can be found in the example source file DrumsClMain.cpp.
In the first step, we create an instance of the `DRUMS` class and initialize:

```
zplane::Drums drums;

error = drums.initialize(inputFile.GetSampleRate(), inputFile.GetNumOfChannels());
```

We then read chunks of data from our input file,

```
while (readNextFrame)
```

And push each chunk into our preProcess() function.

After the entire file has been read and pushed into `DRUMS`, we call finishPre↩ Process() once to terminate the preprocessing stage

We then call the process function:

We can access the individual elements of the resulting drum transcription and print them on the command line:

1.4 Support

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

zplane.development
Goerzallee 311
d-14167 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 Drums

6 Class Documentation

6.1 zplane::Drums::DrumResultElement Struct Reference

`#include <Drums/Drums.h>`

Public Attributes

- float startTimeInS
- float endTimeInS
- std::size_t classIndex
- std::size_t velocity

6.1.1 Detailed Description

Struct representing drum hit events in a result
Definition at line 34 of file Drums.h.

6.1.2 Member Data Documentation

classIndex `std::size_t zplane::Drums::DrumResultElement::classIndex`
Definition at line 38 of file Drums.h.

endTimeInS `float zplane::Drums::DrumResultElement::endTimeInS`
Definition at line 37 of file Drums.h.

startTimeInS `float zplane::Drums::DrumResultElement::startTimeInS`
Definition at line 36 of file Drums.h.

velocity `std::size_t zplane::Drums::DrumResultElement::velocity`
Definition at line 39 of file Drums.h.
The documentation for this struct was generated from the following file:

- Drums/Drums.h

6.2 zplane::Drums Class Reference

`#include <Drums/Drums.h>`

Classes

- struct DrumResultElement

Public Types

- enum ErrorType {
  noError, memError, notInitializedError, alreadyInitializedError,
  invalidFunctionParamError, invalidFunctionCallError, canNotBeCalledAtThis↩
  StageError, preProcessNeverCalledError,
  unfinishedPreProcessingError, unknownError, numErrorTypes }

Public Member Functions

- Drums ()
- ~Drums ()
- ErrorType initialize (float sampleRate, std::size_t numChannels)
- ErrorType preProcess (float const *const *const inputBuffer, int numberOf↩
  Frames)
- ErrorType finishPreProcess ()
- ErrorType process (std::vector< DrumResultElement > &result)
- ErrorType reset ()
- std::size_t getFramesNeeded ()
- std::size_t getMaxFramesNeeded ()
- bool isInitialized ()

Static Public Member Functions

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

6.2.1 Detailed Description

Definition at line 6 of file Drums.h.

6.2.2 Member Enumeration Documentation

ErrorType `enum zplane::Drums::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

Enumerator

canNotBeCalledAtThisStageError	further audio can't be added via preProcess if finishPreProcessing was called with flushBuffers=true or a data chunk was set, finishPreProcessing can't be called if a data chunk was set
preProcessNeverCalledError	finishPreProcessing() can only be called if preProcess has at least been called once
unfinishedPreProcessingError	finishPreProcessing() has to be called before process() or setKnownDownbeat()
unknownError	unknown error occurred
numErrorTypes	

Definition at line 9 of file Drums.h.

```
10         {
11             noError,
12             memError,
13             notInitializedError,
14             alreadyInitializedError,
15             invalidFunctionParamError,
16             invalidFunctionCallError,
17             canNotBeCalledAtThisStageError,
18             preProcessNeverCalledError,
19             unfinishedPreProcessingError,
20             unknownError,
21             numErrorTypes
22         };
```

6.2.3 Constructor & Destructor Documentation

Drums() `zplane::Drums::Drums ( )`

~Drums() `zplane::Drums::~Drums ( )`

6.2.4 Member Function Documentation

finishPreProcess() `ErrorType zplane::Drums::finishPreProcess ( )`

Terminates the preprocessing stage.

Needs to be called once before process() can be called.

Returns

Drums::ErrorType : Returns an error code.

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

Returns the build date string.

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

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

Returns

size_t : required number of sample frames

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

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

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

initialize()
```
ErrorType zplane::Drums::initialize (
    float sampleRate,
    std::size_t numChannels )
```

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

Parameters

sampleRate	Sample rate of the input signal in Hertz.
numChannels	Number of channels in the input signal.

Returns

Drums::ErrorType : Returns an error code.

isInitialized() `bool zplane::Drums::isInitialized ( )`

preProcess()
```
ErrorType zplane::Drums::preProcess (
    float const *const *const inputBuffer,
    int numberOfFrames )
```

Preprocesses a block of audio.

This function can be called multiple times in order to provide successive chunks of the input audio signal.

Parameters

inputBuffer	pointer to the input data chunk. inputBuffer[i] points to the i-th audio channel.
numberOfFrames	The number of audio samples in each audio channel of the provided input data chunk.

Returns

Drums::ErrorType : Returns an error code.

process() `ErrorType zplane::Drums::process ( std::vector< DrumResultElement > & result )`

Perform Drums estimation. This is where the major chunk of computational load happens.

Parameters

result	this array contains the transcription result. The order of the elements is determined by the struct element "startTimeInS".

Returns

Drums::ErrorType : Returns an error code.

reset() `ErrorType zplane::Drums::reset ( )`

Resets the state of the SDK to the same state as directly after calling initialize()

Returns

Drums::ErrorType : Returns an error code.

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

- Drums/Drums.h

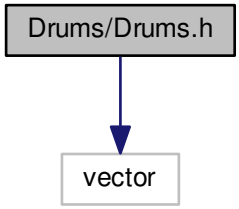
7 File Documentation

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

7.2 Drums/Drums.h File Reference

```
#include <vector>
```

Include dependency graph for Drums.h:



Classes

- class zplane::Drums
- struct zplane::Drums::DrumResultElement

Namespaces

- zplane

Index