TONART 3.0.3

by zplane.development

February 13, 2018

Contents

1 TONART key detection SDK Documentation

1.1 Introduction

TONART key detection is a Software Developer Kit (SDK) that enables the user to automatically determine the key and the concert pitch of any given audio input signal. TONART key detection comes with a C++ API.
The SDK is available for all common operating systems. Apart from the delivered libraries, no additional libraries are required. The SDK is delivered with the libraries, the corresponding header files, this documentation and a small example application.
The first part of this document explains the API of the SDK. Subsequently, a small example explaining the basic usage of the SDK is provided.

1.2 What's new in V3

- Completely redesigned detection algorithm
- Separate analysis modes for different musical genres
- Highly improved accuracy for electronic dance music tracks
- Improved tuning frequeny estimation
- Simpler API
- Renaming of root pitch names to match the most common keys in the circle of fifths (see Naming Conventions)

1.3 API Documentation

1.3.1 Naming Conventions

A **frame** denotes the number of audio samples per channel, i.e. 512 stereo frames correspond to 1024 float values (samples).

A **pitch class** describes the name of a pitch independent of the octave it occurred in. In other words, notes with pitches at C0, C1, C2, etc. all belong to pitch class C.

The **root pitch class** refers to the pitch class of the root of the key, i.e. the root pitch of a C major key is pitch class C. TONART only uses those root pitches that correspond to the most common major keys in the circle of fifths (from 6 ♭ to 5 ♯).

A **key type** can be either *major* or *minor* and denotes the group of pitch classes a

music track is based on. It is also known as tonality or gender. The characteristic interval of a major key is the major third above the root pitch class. Minor keys contain a minor third above the root pitch class.

1.3.2 Required Functions

The following functions have to be called when using the TONART library:

- **ErrorType tONaRTIf::createInstance (tONaRTIf*& pInstancePointer, float fSampleRate, int iNumberOfChannels)**
  Creates a new instance of tONaRT. The handle to the new instance is returned in parameter pInstancePointer. fSampleRate and iNumberOfChannels denote the input samplerate and number of channels, respectively.
  The function returns tONaRTIf::noError upon success. A call to this function is mandatory.

- **void tONaRTIf::destroyInstance (tONaRTIf*& pInstancePointer)**
  Destroys the tONaRT instance provided by pInstancePointer. A call to this function is mandatory.

- **ErrorType tONaRTIf::process (float const *const *const ppfInputBuffer, int iNumberOfFrames)**
  Does the actual pitch analysis. ppfInputBuffer is an array of pointers to the audio data. ppfInputBuffer[0] is a pointer to the data of the first channel, ppfInputBuffer[1] points to the data of the second channel etc. iNumberOfFrames 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. Each chunk must not exceed the maximum number of input frames as specified by tONaRTIf::getMaximumNumberOfInputFrames().

- **tONaRTResultIf tONaRTIf::computeAndGetResult (GenreType eGenre)**
  Returns the result as a tONaRTResultIf object. The parameter eGenre specifies the genre of the audio input. This function can be called anytime after instance creation in between calls to tONaRTIf::process().

1.3.3 Command Line Example

The command line example can be executed by the following command

```
tONaRTCl -i <inputFile> -r <keyResultFile>
```

The complete code can be found in the example source file tONaRTClMain.cpp.
In the first step, a declare a tONaRTIf pointer and create and instance of the tONaRtIf class:

```
tONaRTIf* pInstanceHandle = 0;
eError = tONaRTIf::createInstance (pInstanceHandle, inputFile.GetSampleRate(),
  inputFile.GetNumOfChannels());
```

We then read chunks of data from the input file,

```
bool bReadNextFrame (true);
int iNumFramesRead (0);
int iCurrentFrame (0);

while (bReadNextFrame)
{
    // read audio data
    iNumFramesRead = inputFile.Read (ppfInput, kBlockSize);

    // if there are not enough samples, fill the rest with zeros and stop the loop in the next run
    if(iNumFramesRead < kBlockSize)
    {
        for (int ch = 0; ch < inputFile.GetNumOfChannels(); ch++)
            memset(&ppfInput[ch][iNumFramesRead], 0, (kBlockSize-iNumFramesRead)*sizeof(float));
        bReadNextFrame = false;
    }
```

and push each chunk into the process function

```
    eError = pInstanceHandle->process (ppfInput, iNumFramesRead);
    }
```

After the entire file has been read and pushed into tONaRT (or at any stage after a process call), we can obtain the result by calling computeResult().

```
tONaRTResultIf result = pInstanceHandle->computeResult (
  tONaRTIf::general);
```

We can then display the results or write it to a file

```
cout << "****************************************" << endl;
cout << "Results:" << endl;
cout << "Tuning frequency: " << result.getTuningFrequencyInHz() << " Hz" << endl;
cout << "Key: " << result.getKeyResult().getKey().getName() << " (probability:
   " << result.getKeyResult().getProbability() << ")" << endl << endl;

// write to file
if (bWriteResultToFile)
{
    std::ofstream resultFile;
    resultFile.open (resultPath.c_str());
    resultFile << "tuningFrequency=" << result.getTuningFrequencyInHz() << endl
               << "key=" << result.getKeyResult().getKey().
  getName();
    resultFile.close();
}
```

Finally we can destroy the tONaRTIf instance

```
    tONaRTIf::destroyInstance (pInstanceHandle);
```

1.4 Background Information

While TONART key detection is robust across different musical genres etc., the algorithm expects the input signal to be homogenous, i.e. to contain only one key without (too much) modulations.

1.5 Support

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

zplane.development GmbH & Co KG
grunewaldstr. 83
d-10823 berlin
germany

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

@: info@zplane.de

2 Class Index

2.1 Class List

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

3 File Index

3.1 File List

Here is a list of all files with brief descriptions:

4 Class Documentation

4.1 tONaRTResultIf::Key Class Reference

`#include <tONaRTResultIf.h>`

Collaboration diagram for tONaRTResultIf::Key:

Public Types

- enum RootPitchClass {
  C, DFlat, D, EFlat,
  E, F, GFlat, G,
  AFlat, A, BFlat, B }
- enum Type { major, minor }

Public Member Functions

- Key (Impl *pImpl)
- Key (const Key &)
- ~Key ()
- RootPitchClass getRootPitchClass () const
- Type getType () const
- std::string getName () const

Static Public Attributes

- static const int iNumRootPitchClasses = 12
- static const int iNumKeyTypes = 2

4.1.1 Detailed Description

Key class.

4.1.2 Member Enumeration Documentation

RootPitchClass `enum tONaRTResultIf::Key::RootPitchClass`

Root pitch classes

Enumerator

C	
DFlat	
D	
EFlat	
E	
F	
GFlat	
G	
AFlat	
A	
BFlat	
B	

Type `enum tONaRTResultIf::Key::Type`

Key types

Enumerator

major	
minor	

4.1.3 Constructor & Destructor Documentation

Key() [1/2] `tONaRTResultIf::Key::Key ( Impl * pImpl )`

Key() [2/2] `tONaRTResultIf::Key::Key ( const Key & )`

Copy constructor.

~Key() `tONaRTResultIf::Key::~Key ( )`

Destructor.

4.1.4 Member Function Documentation

getName() `std::string tONaRTResultIf::Key::getName ( ) const`

Returns the key string in the format <rootPitch>:<keyType>.

getRootPitchClass() `RootPitchClass tONaRTResultIf::Key::getRootPitchClass ( ) const`

Returns the root pitch class.

getType() `Type tONaRTResultIf::Key::getType ( ) const`

Returns the key type.

4.1.5 Member Data Documentation

iNumKeyTypes `const int tONaRTResultIf::Key::iNumKeyTypes = 2 [static]`

iNumRootPitchClasses `const int tONaRTResultIf::Key::iNumRootPitchClasses = 12 [static]`

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

- tONaRTResultIf.h

4.2 tONaRTResultIf::KeyResult Class Reference

`#include <tONaRTResultIf.h>`

Collaboration diagram for tONaRTResultIf::KeyResult:

Public Member Functions

- KeyResult (Impl *pImpl)
- KeyResult (const KeyResult &)
- KeyResult & operator= (const KeyResult &)
- ~KeyResult ()
- Key getKey () const
- float getProbability () const

4.2.1 Detailed Description

Key result class for accessing the key and its probability.

4.2.2 Constructor & Destructor Documentation

KeyResult() [1/2] `tONaRTResultIf::KeyResult::KeyResult ( Impl * pImpl )`

KeyResult() [2/2] `tONaRTResultIf::KeyResult::KeyResult ( const KeyResult & )`

Copy constructor.

~KeyResult() `tONaRTResultIf::KeyResult::~KeyResult ( )`

Destructor.

4.2.3 Member Function Documentation

getKey() `Key tONaRTResultIf::KeyResult::getKey ( ) const`

Returns the key.

getProbability() `float tONaRTResultIf::KeyResult::getProbability ( ) const`

Returns the probability.

operator=() `KeyResult& tONaRTResultIf::KeyResult::operator= ( const KeyResult & )`

Assignment operator.

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

- tONaRTResultIf.h

4.3 tONaRTIf Class Reference

`#include <tONaRTIf.h>`

Collaboration diagram for tONaRTIf:

Public Types

- enum ErrorType { noError, memError, invalidFunctionParamError, invalidFunctionCallError }
- enum VersionType { major, minor, patch, revision }
- enum GenreType { electronic, general, numGenreTypes }

Public Member Functions

- virtual int getMaximumNumberOfInputFrames ()=0
- virtual ErrorType process (float const *const *const ppfInputBuffer, int iNumberOfFrames)=0
- virtual tONaRTResultIf computeResult (GenreType eGenre)=0

Static Public Member Functions

- static const int getVersion (const VersionType eVersionIdx)
- static std::string getBuildDate ()
- static ErrorType createInstance (tONaRTIf *&pInstancePointer, float fSampleRate, int iNumberOfChannels)
- static void destroyInstance (tONaRTIf *&pInstancePointer)

Protected Member Functions

- virtual ~tONaRTIf ()

4.3.1 Member Enumeration Documentation

ErrorType `enum tONaRTIf::ErrorType`

Error codes

Enumerator

noError	no error occurred
memError	memory allocation failed
invalidFunctionParamError	one or more function parameters are not valid
invalidFunctionCallError	function call not allowed at this stage

GenreType `enum tONaRTIf::GenreType`

Musical genre

Enumerator

electronic	electronic dance music
general	general purpose for all other musical styles
numGenreTypes	

VersionType `enum tONaRTIf::VersionType`

Version number

Enumerator

major	major version number
minor	minor version number
patch	patch version number
revision	revision number

4.3.2 Constructor & Destructor Documentation

~tONaRTIf() `virtual tONaRTIf::~tONaRTIf ( ) [inline], [protected], [virtual]`

4.3.3 Member Function Documentation

computeResult()
```
virtual tONaRTResultIf tONaRTIf::computeResult (
        GenreType eGenre ) [pure virtual]
```
Returns the results until this point of the analysis.

createInstance()
```
static ErrorType tONaRTIf::createInstance (
        tONaRTIf *& pInstancePointer,
        float fSampleRate,
        int iNumberOfChannels ) [static]
```
Creates a new instance of [tONaRT] key detection.

destroyInstance()
```
static void tONaRTIf::destroyInstance (
        tONaRTIf *& pInstancePointer ) [static]
```
Destroys an instance of [tONaRT] key detection.

getBuildDate() `static std::string tONaRTIf::getBuildDate ( ) [static]`
Returns the build date string.

getMaximumNumberOfInputFrames() `virtual int tONaRTIf::getMaximumNumber↩ OfInputFrames ( ) [pure virtual]`
Returns the maximum number of input frames per process call.

getVersion() `static const int tONaRTIf::getVersion (`
`const VersionType eVersionIdx ) [static]`
Returns major version, minor version, patch and build number of this [tONaRT] version.

Parameters

eVersionIdx	requested version number part

Returns

version number part

process() `virtual ErrorType tONaRTIf::process (`
`float const *const *const ppfInputBuffer,`
`int iNumberOfFrames ) [pure virtual]`
Processes a block of audio. iNumberOfFrames must not exceed the maximum number of frames as returned by GetMaximumNumberOfInputFrames().
The documentation for this class was generated from the following file:

- tONaRTIf.h

4.4 tONaRTResultIf Class Reference

`#include <tONaRTResultIf.h>`
Collaboration diagram for tONaRTResultIf:

Classes

- class Key
- class KeyResult

Public Member Functions

- tONaRTResultIf (Impl *pImpl)
- tONaRTResultIf (const tONaRTResultIf &)
- virtual ~tONaRTResultIf ()
- tONaRTResultIf & operator= (const tONaRTResultIf &)
- float getTuningFrequencyInHz () const
- KeyResult getKeyResult () const
- std::vector< KeyResult > getAllKeyResults () const

4.4.1 Constructor & Destructor Documentation

tONaRTResultIf() [1/2] `tONaRTResultIf::tONaRTResultIf ( Impl * pImpl )`

tONaRTResultIf() [2/2] `tONaRTResultIf::tONaRTResultIf ( const tONaRTResultIf & )`

Copy constructor.

~tONaRTResultIf() `virtual tONaRTResultIf::~tONaRTResultIf ( ) [virtual]`

Destructor.

4.4.2 Member Function Documentation

getAllKeyResults() `std::vector<KeyResult> tONaRTResultIf::getAllKeyResults ( ) const`

Returns a vector of key results in descending order of their probability.

getKeyResult() `KeyResult tONaRTResultIf::getKeyResult ( ) const`

Returns the most likely key.

getTuningFrequencyInHz() `float tONaRTResultIf::getTuningFrequencyInHz ( ) const`

Returns the estimated tuning frequency.

operator=() `tONaRTResultIf& tONaRTResultIf::operator= ( const tONaRTResultIf & )`

Assignment operator.

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

- tONaRTResultIf.h

5 File Documentation

5.1 docugen.txt File Reference

5.1.1 Detailed Description

source documentation main file

5.2 tONaRTIf.h File Reference

interface of the tONaRTIf class.

```
#include <string>
```

Include dependency graph for tONaRTIf.h:

5.3 tONaRTResultIf.h File Reference

interface of the tONaRTResultIf class.

```
   #include <string>
#include <vector>
```

Include dependency graph for tONaRTResultIf.h:

Classes

- class tONaRTResultIf
- class tONaRTResultIf::Key
- class tONaRTResultIf::KeyResult

5.3.1 Detailed Description

: