aubio Documentation

Release 0.4.6

Paul Brossier

Oct 04, 2017

Contents

aubio is a collection of algorithms and tools to label and transform music and sounds. It scans or *listens* to audio signals and attempts to detect musical events. For instance, when a drum is hit, at which frequency is a note, or at what tempo is a rhythmic melody.

aubio features include segmenting a sound file before each of its attacks, performing pitch detection, tapping the beat and producing midi streams from live audio.

CHAPTER 1

Quick links

- *Python module*
- *Command line tools*
- *Developping with aubio*
- *Building aubio*

CHAPTER 2

Project pages

- Project homepage: https://aubio.org
- aubio on github: https://github.com/aubio/aubio
- aubio on pypi: https://pypi.python.org/pypi/aubio
- Doxygen documentation: https://aubio.org/doc/latest/
- Mailing lists: https://lists.aubio.org
- Travis Continuous integration page
- Appveyor Continuous integration page
- Landscape python code validation
- ReadTheDocs documentation

CHAPTER 3

Features

aubio provides several algorithms and routines, including:

- several onset detection methods
- different pitch detection methods
- tempo tracking and beat detection
- MFCC (mel-frequency cepstrum coefficients)
- FFT and phase vocoder
- up/down-sampling
- digital filters (low pass, high pass, and more)
- spectral filtering
- transient/steady-state separation
- sound file read and write access
- various mathematics utilities for music applications

The name aubio comes from *audio* with a typo: some errors are likely to be found in the results.

CHAPTER 4

Content

Installing aubio

aubio runs on Linux, Windows, macOS, iOS, Android, and probably a few others operating systems.

Aubio is available as a C library and as a python module.

Cheat sheet

- *get aubio latest source code*:

```
# official repo
git clone https://git.aubio.org/aubio/aubio
# mirror
git clone https://github.com/aubio/aubio
# latest release
wget https://aubio.org/pub/aubio-<version>.tar.gz
```

- *build aubio from source*:

```
# 1. simple
cd aubio
make

# 2. step by step
./scripts/get_waf.sh
./waf configure
./waf build
sudo ./waf install
```

- *install python-aubio from source*:

```
# from git
pip install git+https://git.aubio.org/aubio/aubio/
# mirror
pip install git+https://github.com/aubio/aubio/
# from latest release
pip install https://aubio.org/pub/aubio-latest.tar.bz2
# from pypi
pip install aubio
# from source directory
cd aubio
pip install -v .
```

- *install python-aubio from a pre-compiled binary*:

```
# conda [osx, linux, win]
conda install -c conda-forge aubio
# .deb (debian, ubuntu) [linux]
sudo apt-get install python3-aubio python-aubio aubio-tools
# brew [osx]
brew install aubio --with-python
```

- *get a pre-compiled version of libaubio*:

```
# .deb (linux) WARNING: old version
sudo apt-get install aubio-tools

# python module
./setup.py install
# using pip
pip install .
```

- *check the list of optional dependencies*:

```
# debian / ubuntu
dpkg -l libavcodec-dev libavutil-dev libavformat-dev \
        libswresample-dev libavresample-dev \
        libsamplerate-dev libsndfile-dev \
        txt2man doxygen
```

Downloading aubio

A number of distributions already include aubio. Check your favorite package management system, or have a look at the aubio download page for more options.

To use aubio in an android project, see *Android build*.

To compile aubio from source, read *Building aubio*.

Pre-compiled binaries

Pre-compiled binaries are available for macOS, iOS, and windows

To use aubio in a macOS or iOS application, see *Frameworks for Xcode*.

Debian/Ubuntu packages

For the latest Debian packages, see https://packages.debian.org/src:aubio.

For the latest Ubuntu packages, see http://packages.ubuntu.com/src:aubio.

For the latest version of the packages, see https://anonscm.debian.org/cgit/collab-maint/aubio.git/. Use `git-buildpackage` to build from the git repository. For instance:

```
$ git clone git://anonscm.debian.org/collab-maint/aubio.git
$ cd aubio
$ git buildpackage
```

Building aubio

Note: To download a prebuilt version of aubio, see *Downloading aubio*.

aubio uses waf to configure, compile, and test the source. A copy of waf is included in aubio tarball, so all you need is a terminal, a compiler, and a recent version of python installed.

Note: Make sure you have all the *Build options* you want before building.

Latest release

The **latest stable release** can be downloaded from https://aubio.org/download:

```
$ curl -O http://aubio.org/pub/aubio-<version>.tar.bz2
$ tar xf aubio-<version>.tar.bz2
$ cd aubio-<version>/
```

Git repository

The **latest git branch** can be obtained with:

```
$ git clone git://git.aubio.org/git/aubio
$ cd aubio/
```

The following command will fetch the correct waf version (not included in aubio's git):

```
$ ./scripts/get_waf.sh
```

Note: Windows users without Git Bash installed will want to use the following commands instead:

```
$ curl -fsS -o waf https://waf.io/waf-1.8.22
$ curl -fsS -o waf.bat https://raw.githubusercontent.com/waf-project/waf/master/utils/
↪waf.bat
```

Compiling

To compile the C library, examples programs, and tests, run:

```
$ ./waf configure
```

Check out the available options using `./waf configure --help`. Once you are done with configuration, you can start building:

```
$ ./waf build
```

To install the freshly built C library and tools, simply run the following command:

```
$ sudo ./waf install
```

Note: Windows users should simply run `waf`, without the leading `./`. For instance:

```
$ waf configure build
```

Running as a user

To use aubio without actually installing, for instance if you don't have root access to install libaubio on your system,

On Linux or macOS, sourcing the script `scripts/setenv_local.sh` should help:

```
$ source ./scripts/setenv_local.sh
```

This script sets `LD_LIBRARY_PATH`, for libaubio, and `PYTHONPATH` for the python module.

On Linux, you should be able to set `LD_LIBRARY_PATH` with:

```
$ export LD_LIBRARY_PATH=$LD_LIBRARY_PATH:$PWD/build/src
```

On Mac OS X, a copy or a symlink can be made in `~/lib`:

```
$ mkdir -p ~/lib
$ ln -sf $PWD/build/src/libaubio*.dylib ~/lib/
```

Note on Mac OS X systems older than El Capitan (10.11), the `DYLD_LIBRARY_PATH` variable can be set as follows:

```
$ export DYLD_LIBRARY_PATH=$DYLD_LIBRARY_PATH:$PWD/build/src
```

Cleaning

If you wish to uninstall the files installed by the `install` command, use `uninstall`:

```
$ sudo ./waf uninstall
```

To clean the source directory, use the `clean` command:

```
$ ./waf clean
```

To also forget the options previously passed to the last `./waf configure` invocation, use the `distclean` command:

```
$ ./waf distclean
```

Frameworks for Xcode

Binary frameworks are available and ready to use in your XCode project, for iOS and macOS.

1. Download and extract the corresponding `framework.zip` file from the *Download* page
2. Select **Build Phases** in your project setting and unfold **Link Binary with Libraries**
3. Add *AudioToolbox* and *Accelerate* system frameworks (or make sure they are listed)
4. Add `aubio.framework` from the unzipped `framework.zip`
5. Include the aubio header in your code:
   - in C/C++:

```
#include <aubio/aubio.h>
```

   - in Obj-C:

```
#import <aubio/aubio.h>
```

   - in Swift:

```
import aubio
```

Using aubio from swift

Once you have downloaded and installed *aubio.framework*, you sould be able to use aubio from C, Obj-C, and Swift source files.

Here is a short example showing how to read a sound file in swift:

```
import aubio

let path = Bundle.main.path(forResource: "example", ofType: "mp4")
if (path != nil) {
    let hop_size : uint_t = 512
    let a = new_fvec(hop_size)
    let b = new_aubio_source(path, 0, hop_size)
    var read: uint_t = 0
    var total_frames : uint_t = 0
    while (true) {
        aubio_source_do(b, a, &read)
        total_frames += read
        if (read < hop_size) { break }
    }
    print("read", total_frames, "frames at", aubio_source_get_samplerate(b),␣
↪"Hz")
    del_aubio_source(b)
    del_fvec(a)
} else {
```

```
    print("could not find file")
}
```

Android build

To compile aubio for android, you will need to get the Android Native Development Toolkit (NDK), prepare a standalone toolchain, and tell waf to use the NDK toolchain. An example script to complete these tasks is available in `scripts/build_android`.

Build options

If built without any external dependencies aubio can be somewhat useful, for instance to read, process, and write simple wav files.

To support more media input formats and add more features to aubio, you can use one or all of the following *external libraries*.

You may also want to know more about the *other options* and the *platform notes*

The configure script will automatically for these extra libraries. To make sure the library or feature is used, pass the *–enable-flag* to waf. To disable this feature, use *–disable-feature*.

To find out more about the build commands, use the *–verbose* option.

External libraries

External libraries are checked for using `pkg-config`. Set the `PKG_CONFIG_PATH` environment variable if you have them installed in an unusual location.

Note: If `pkg-config` is not found in `PATH`, the configure step will succeed, but none of the external libraries will be used.

---

libav

libav.org, open source audio and video processing tools.

If all of the following libraries are found, they will be used to compile `aubio_source_avcodec`. so that `aubio_source` will be able to decode audio from all formats supported by libav.

- libavcodec
- libavformat
- libavutil
- libavresample

To enable this option, configure with `--enable-avcodec`. The build will then failed if the required libraries are not found. To disable this option, configure with `--disable-avcodec`

libsndfile

libsndfile, a C library for reading and writing sampled sound files.

With libsndfile built in, `aubio_source_sndfile` will be built in and used by `aubio_source`.

To enable this option, configure with `--enable-sndfile`. The build will then fail if the required library is not found. To disable this option, configure with `--disable-sndfile`

libsamplerate

libsamplerate, a sample rate converter for audio.

With libsamplerate built in, `aubio_source_sndfile` will support resampling, and `aubio_resample` will be fully functional.

To enable this option, configure with `--enable-samplerate`. The build will then fail if the required library is not found. To disable this option, configure with `--disable-samplerate`

libfftw3

FFTW, a C subroutine for computing the discrete Fourier transform

With libfftw3 built in, `aubio_fft` will use FFTW to compute Fast Fourier Transform (FFT), allowing aubio to compute FFT on length that are not a power of 2.

To enable this option, configure with `--enable-fftw3`. The build will then fail if the required library is not found. To disable this option, configure with `--disable-fftw3`

Platform notes

On all platforms, you will need to have installed:

- a compiler (gcc, clang, msvc, ...)
- python (any version >= 2.7, including 3.x)
- a terminal to run command lines in

Linux

The following *External libraries* will be used if found: libav, libsamplerate, libsndfile, *libfftw3*.

macOS

The following system frameworks will be used on Mac OS X systems:

- Accelerate to compute FFTs and other vectorized operations optimally.
- CoreAudio and AudioToolbox to decode audio from files and network streams.

Note: To build a fat binary for both `i386` and `x86_64`, use `./waf configure --enable-fat`.

The following *External libraries* will also be checked: libav, libsamplerate, libsndfile, *libfftw3*.

To build a fat binary on a darwin like system (macOS, tvOS, appleOS, ...) platforms, configure with `--enable-fat`.

Windows

To use a specific version of the compiler, `--msvc_version`. To build for a specific architecture, use `--msvc_target`. For instance, to build aubio for `x86` using `msvc 12.0`, use:

```
waf configure --msvc_version='msvc 12.0' --msvc_target='x86'
```

The following *External libraries* will be used if found: libav, libsamplerate, libsndfile, *libfftw3*.

iOS

The following system frameworks will be used on iOS and iOS Simulator.

- Accelerate to compute FFTs and other vectorized operations optimally.
- CoreAudio and AudioToolbox to decode audio from files and network streams.

To build aubio for iOS, configure with `--with-target-platform=ios`. For the iOS Simulator, use `--with-target-platform=iosimulator` instead.

By default, aubio is built with the following flags on iOS:

```
CFLAGS="-fembed-bitcode -arch arm64 -arch armv7 -arch armv7s -miphoneos-version-min=6.1"
```

and on iOS Simulator:

```
CFLAGS="-arch i386 -arch x86_64 -mios-simulator-version-min=6.1"
```

Set `CFLAGS` and `LINKFLAGS` to change these default values, or edit `wscript` directly.

Other options

Some additional options can be passed to the configure step. For the complete list of options, run:

```
$ ./waf --help
```

Here is an example of a custom command:

```
$ ./waf --verbose configure build install \
            --enable-avcodec --enable-wavread --disable-wavwrite \
            --enable-sndfile --enable-samplerate --enable-docs \
            --destdir $PWD/build/destdir --testcmd="echo %s" \
            --prefix=/opt --libdir=/opt/lib/multiarch \
            --manpagesdir=/opt/share/man  \
            uninstall clean distclean dist distcheck
```

Double precision

To compile aubio in double precision mode, configure with `--enable-double`.

To compile aubio in single precision mode, use `--disable-double` (default).

Disabling the tests

In some case, for instance when cross-compiling, unit tests should not be run. Option `--notests` can be used for this purpose. The tests will not be executed, but the binaries will be compiled, ensuring that linking against libaubio works as expected.

Note: The `--notests` option should be passed to both `build` and `install` targets, otherwise waf will try to run them.

Edit wscript

Many of the options are gathered in the file *wscript*. a good starting point when looking for additional options.

Building the docs

If the following command line tools are found, the documentation will be built built:

- doxygen to build the *Doxygen documentation*.
- txt2man to build the *Command line tools*
- sphinx to build this document

These tools are searched for in the current `PATH` environment variable. By default, the documentation is built only if the tools are found.

To disable the documentation, configure with `--disable-docs`. To build with the documentation, configure with `--enable-docs`.

Python module

The aubio extension for Python is available for Python 2.7 and Python 3.

Installing aubio with pip

aubio can now be installed using `pip`:

```
$ pip install aubio
```

Building the module

From `aubio` source directory, run the following:

```
$ ./setup.py clean
$ ./setup.py build
$ sudo ./setup.py install
```

Using aubio in python

Once you have python-aubio installed, you should be able to run `python -c "import aubio; print(aubio.version)"`.

A simple example

Here is a `simple script` that reads all the samples from a media file:

```
#! /usr/bin/env python
import sys, aubio

samplerate = 0  # use original source samplerate
hop_size = 256 # number of frames to read in one block
s = aubio.source(sys.argv[1], samplerate, hop_size)
total_frames = 0

while True: # reading loop
    samples, read = s()
    total_frames += read
    if read < hop_size: break # end of file reached

fmt_string = "read {:d} frames at {:d}Hz from {:s}"
print (fmt_string.format(total_frames, s.samplerate, sys.argv[1]))
```

Filtering an input sound file

Here is a more complete example, `demo_filter.py`. This files executes the following:

- read an input media file (`aubio.source`)
- filter it using an A-weighting filter (`aubio.digital_filter`)
- write result to a new file (`aubio.sink`)

```
#! /usr/bin/env python


def apply_filter(path):
    from aubio import source, sink, digital_filter
    from os.path import basename, splitext

    # open input file, get its samplerate
    s = source(path)
    samplerate = s.samplerate

    # create an A-weighting filter
    f = digital_filter(7)
    f.set_a_weighting(samplerate)
    # alternatively, apply another filter

    # create output file
    o = sink("filtered_" + splitext(basename(path))[0] + ".wav", samplerate)

    total_frames = 0
    while True:
```

```
        samples, read = s()
        filtered_samples = f(samples)
        o(filtered_samples, read)
        total_frames += read
        if read < s.hop_size: break

    duration = total_frames / float(samplerate)
    print ("read {:s}".format(s.uri))
    print ("applied A-weighting filtered ({:d} Hz)".format(samplerate))
    print ("wrote {:s} ({:.2f} s)".format(o.uri, duration))

if __name__ == '__main__':
    import sys
    for f in sys.argv[1:]:
        apply_filter(f)
```

More demos

Check out the python demos folder for more examples.

Python tests

A number of python tests are provided. To run them, use `python/tests/run_all_tests`.

Command line tools

The python module comes with the following tools:

- `aubio` estimate and extract descriptors from sound files
- `aubiocut` slices sound files at onset or beat timestamps

More command line tools are included along with the library.

- `aubioonset` outputs the time stamp of detected note onsets
- `aubiopitch` attempts to identify a fundamental frequency, or pitch, for each frame of the input sound
- `aubiomfcc` computes Mel-frequency Cepstrum Coefficients
- `aubiotrack` outputs the time stamp of detected beats
- `aubionotes` emits midi-like notes, with an onset, a pitch, and a duration
- `aubioquiet` extracts quiet and loud regions

aubio

```
NAME
  aubio - a command line tool to extract information from sound files

SYNOPSIS

  aubio [-h] [-V] <command> ...
```

```
COMMANDS

  The general syntax is "aubio <command> <soundfile> [options]". The following
  commands are available:

  onset        get onset times
  pitch        extract fundamental frequency
  beat         get locations of beats
  tempo        get overall tempo in bpm
  notes        get midi-like notes
  mfcc         extract mel-frequency cepstrum coefficients
  melbands     extract mel-frequency energies per band

  For a list of available commands, use "aubio -h". For more info about each
  command, use "aubio <command> --help".

GENERAL OPTIONS

  These options can be used before any command has been specified.

  -h, --help  show help message and exit

  -V, --version  show version

COMMON OPTIONS

  The following options can be used with all commands:

  <source_uri>, -i <source_uri>, --input <source_uri>  input sound file to
  analyse (required)

  -r <freq>, --samplerate <freq>  samplerate at which the file should be
  represented (default: 0, e.g. samplerate of the input sound)

  -H <size>, --hopsize <size>  overlap size, number of samples between two
  consecutive analysis (default: 256)

  -B <size>, --bufsize <size>  buffer size, number of samples used for each
  analysis, (e.g. FFT length, default: 512)

  -h, --help  show help message and exit

  -T format, --time-format format  select time values output format (samples,
  ms, seconds) (default: seconds)

  -v, --verbose  be verbose (increment verbosity by 1, default: 1)

  -q, --quiet  be quiet (set verbosity to 0)

ONSET

  The following additional options can be used with the "onset" subcommand.

  -m <method>, --method <method>  onset novelty function
  <default|energy|hfc|complex|phase|specdiff|kl|mkl|specflux> (default:
  default)

  -t <threshold>, --threshold <threshold>  threshold (default: unset)
```

```
  -s <value>, --silence <value>  silence threshold, in dB (default: -70)

  -M <value>, --minioi <value>  minimum Inter-Onset Interval (default: 12ms)

PITCH

  The following additional options can be used with the "pitch" subcommand.

  -m <method>, --method <method>  pitch detection method
  <default|yinfft|yin|mcomb|fcomb|schmitt> (default: default, e.g. yinfft)

  -t <threshold>, --threshold <threshold>  tolerance (default: unset)

  -s <value>, --silence <value>  silence threshold, in dB (default: -70)

  The default buffer size for the beat algorithm is 2048. The default hop size
  is 256.

BEAT

  The "beat" command accepts all common options and no additional options.

  The default buffer size for the beat algorithm is 1024. The default hop size
  is 512.

TEMPO

  The "tempo" command accepts all common options and no additional options.

  The default buffer size for the beat algorithm is 1024. The default hop size
  is 512.

NOTES

  The "note" command accepts all common options and no additional options.

MFCC

  The "mfcc" command accepts all common options and no additional options.

MELBANDS

  The "melbands" command accepts all common options and no additional options.

EXAMPLES

  Extract onsets using a minimum inter-onset interval of 30ms:

    aubio onset /path/to/input_file -M 30ms

  Extract pitch with method "mcomb" and a silence threshold of -90dB:

    aubio pitch /path/to/input_file -m mcomb -s -90.0

  Extract MFCC using the standard Slaney implementation:

    aubio mfcc /path/to/input_file -r 44100
```

```
SEE ALSO

  aubiocut(1)

AUTHOR

  This manual page was written by Paul Brossier <piem@aubio.org>. Permission is
  granted to copy, distribute and/or modify this document under the terms of
  the GNU General Public License as published by the Free Software Foundation,
  either version 3 of the License, or (at your option) any later version.
```

aubiocut

```
NAME
  aubiocut - a command line tool to slice sound files at onset or beat timestamps

SYNOPSIS

  aubiocut source
  aubiocut [[-i] source]
           [-r rate] [-B win] [-H hop]
           [-O method] [-t thres]
           [-b] [-c]
           [-v] [-q] [-h]

OPTIONS

  This program follows the usual GNU command line syntax, with long options
  starting with two dashes (--). A summary of options is included below.

  -i, --input source  Run analysis on this audio file. Most uncompressed and
  compressed are supported, depending on how aubio was built.

  -r, --samplerate rate  Fetch the input source, resampled at the given
  sampling rate. The rate should be specified in Hertz as an integer. If set
  to 0, the sampling rate of the original source will be used. Defaults to 0.

  -B, --bufsize win  The size of the buffer to analyze, that is the length
  of the window used for spectral and temporal computations. Defaults to 512.

  -H, --hopsize hop  The number of samples between two consecutive analysis.
  Defaults to 256.

  -O, --onset method  The onset detection method to use. See ONSET METHODS
  below. Defaults to 'default'.

  -b, --beat  Use beat locations instead of onset locations.

  -t, --onset-threshold thres  Set the threshold value for the onset peak
  picking. Values are typically in the range [0.001, 0.900]. Lower threshold
  values imply more onsets detected. Increasing this threshold should reduce
  the number of incorrect detections. Defaults to 0.3.

  -c, --cut  Cut input sound file at detected labels. A new sound files for
```

```
  each slice will be created in the current directory.

  -o, --output directory  Specify the directory path where slices of the
  original source should be created.

  --cut-until-nsamples n  How many extra samples should be added at the end of
  each slice (default 0).

  --cut-until-nslices n  How many extra slices should be added at the end of
  each slice (default 0).

  -h, --help  Print a short help message and exit.

  -v, --verbose  Be verbose.

  -q, --quiet  Be quiet.


ONSET METHODS

  Available methods: default, energy, hfc, complex, phase, specdiff, kl, mkl,
  specflux.

  See aubioonset(1) for details about these methods.


SEE ALSO

  aubioonset(1),
  aubiopitch(1),
  aubiotrack(1),
  aubionotes(1),
  aubioquiet(1),
  and
  aubiomfcc(1).

AUTHOR

  This manual page was written by Paul Brossier <piem@aubio.org>. Permission is
  granted to copy, distribute and/or modify this document under the terms of
  the GNU General Public License as published by the Free Software Foundation,
  either version 3 of the License, or (at your option) any later version.
```

aubioonset

```
NAME
  aubioonset - a command line tool to extract musical onset times

SYNOPSIS

  aubioonset source
  aubioonset [[-i] source] [-o sink]
             [-r rate] [-B win] [-H hop]
             [-O method] [-t thres]
             [-T time-format]
             [-s sil] [-m] [-f]
```

```
             [-j] [-N miditap-note] [-V miditap-velo]
             [-v] [-h]

DESCRIPTION

  aubioonset attempts to detect onset times, the beginning of discrete sound
  events, in audio signals.

  When started with an input source (-i/--input), the detected onset times are
  given on the console, in seconds.

  When started without an input source, or with the jack option (-j/--jack),
  aubioonset starts in jack mode.

OPTIONS

  This program follows the usual GNU command line syntax, with long options
  starting with two dashes (--). A summary of options is included below.

  -i, --input source  Run analysis on this audio file. Most uncompressed and
  compressed are supported, depending on how aubio was built.

  -o, --output sink  Save results in this file. The file will be created on
  the model of the input file. Onset times are marked by a short wood-block
  like sound.

  -r, --samplerate rate  Fetch the input source, resampled at the given
  sampling rate. The rate should be specified in Hertz as an integer. If 0,
  the sampling rate of the original source will be used. Defaults to 0.

  -B, --bufsize win  The size of the buffer to analyze, that is the length
  of the window used for spectral and temporal computations. Defaults to 512.

  -H, --hopsize hop  The number of samples between two consecutive analysis.
  Defaults to 256.

  -O, --onset method  The onset detection method to use. See ONSET METHODS
  below. Defaults to 'default'.

  -t, --onset-threshold thres  Set the threshold value for the onset peak
  picking. Values are typically in the range [0.001, 0.900]. Lower threshold
  values imply more onsets detected. Increasing this threshold should reduce
  the number of incorrect detections. Defaults to 0.3.

  -M, --minioi value  Set the minimum inter-onset interval, in seconds, the
  shortest interval between two consecutive onsets. Defaults to 0.020

  -s, --silence sil  Set the silence threshold, in dB, under which the onset
  will not be detected. A value of -20.0 would eliminate most onsets but the
  loudest ones. A value of -90.0 would select all onsets. Defaults to -90.0.

  -T, --timeformat format  Set time format (samples, ms, seconds). Defaults to
  seconds.

  -m, --mix-input  Mix source signal to the output signal before writing to
  sink.

  -f, --force-overwrite  Overwrite output file if it already exists.
```

```
  -j, --jack  Use Jack input/output. You will need a Jack connection
  controller to feed aubio some signal and listen to its output.

  -N, --miditap-note  Override note value for MIDI tap. Defaults to 69.

  -V, --miditap-velop  Override velocity value for MIDI tap. Defaults to 65.

  -h, --help  Print a short help message and exit.

  -v, --verbose  Be verbose.

ONSET METHODS

  Available methods are:

  default  Default distance, currently hfc

  Default: 'default' (currently set to hfc)

  energy  Energy based distance

  This function calculates the local energy of the input spectral frame.

  hfc  High-Frequency content

  This method computes the High Frequency Content (HFC) of the input
  spectral frame. The resulting function is efficient at detecting
  percussive onsets.

  Paul Masri. Computer modeling of Sound for Transformation and Synthesis of
  Musical Signal. PhD dissertation, University of Bristol, UK, 1996.

  complex  Complex domain onset detection function

  This function uses information both in frequency and in phase to determine
  changes in the spectral content that might correspond to musical onsets.
  It is best suited for complex signals such as polyphonic recordings.

  Christopher Duxbury, Mike E. Davies, and Mark B. Sandler.  Complex domain
  onset detection for musical signals. In Proceedings of the Digital Audio
  Effects Conference, DAFx-03, pages 90-93, London, UK, 2003.

  phase  Phase based onset detection function

  This function uses information both in frequency and in phase to determine
  changes in the spectral content that might correspond to musical onsets. It
  is best suited for complex signals such as polyphonic recordings.

  Juan-Pablo Bello, Mike P. Davies, and Mark B. Sandler.  Phase-based note
  onset detection for music signals. In Proceedings of the IEEE International
  Conference on Acoustics Speech and Signal Processing, pages 441444,
  Hong-Kong, 2003.

  specdiff  Spectral difference onset detection function

  Jonhatan Foote and Shingo Uchihashi. The beat spectrum: a new approach to
  rhythm analysis. In IEEE International Conference on Multimedia and Expo
```

```
  (ICME 2001), pages 881884, Tokyo, Japan, August 2001.

  kl  Kulback-Liebler onset detection function

  Stephen Hainsworth and Malcom Macleod. Onset detection in music audio
  signals. In Proceedings of the International Computer Music Conference
  (ICMC), Singapore, 2003.

  mkl  Modified Kulback-Liebler onset detection function

  Paul Brossier, ``Automatic annotation of musical audio for interactive
  systems'', Chapter 2, Temporal segmentation, PhD thesis, Centre for
  Digital music, Queen Mary University of London, London, UK, 2006.

  specflux  Spectral flux

  Simon Dixon, Onset Detection Revisited, in ``Proceedings of the 9th
  International Conference on Digital Audio Effects'' (DAFx-06), Montreal,
  Canada, 2006.

SEE ALSO

  aubiopitch(1),
  aubiotrack(1),
  aubionotes(1),
  aubioquiet(1),
  aubiomfcc(1),
  and
  aubiocut(1).

AUTHOR

  This manual page was written by Paul Brossier <piem@aubio.org>. Permission is
  granted to copy, distribute and/or modify this document under the terms of
  the GNU General Public License as published by the Free Software Foundation,
  either version 3 of the License, or (at your option) any later version.
```

aubiopitch

```
NAME
  aubiopitch - a command line tool to extract musical pitch

SYNOPSIS

  aubiopitch source
  aubiopitch [[-i] source] [-o sink]
             [-r rate] [-B win] [-H hop]
             [-p method] [-u unit] [-l thres]
             [-T time-format]
             [-s sil] [-f]
             [-v] [-h] [-j]

DESCRIPTION

  aubiopitch attempts to detect the pitch, the perceived height of a musical
  note.
```

```
  When started with an input source (-i/--input), the detected pitch are
  printed on the console, prefixed by a timestamp in seconds. If no pitch
  candidate is found, the output is 0.

  When started without an input source, or with the jack option (-j/--jack),
  aubiopitch starts in jack mode.

OPTIONS

  This program follows the usual GNU command line syntax, with long options
  starting with two dashes (--). A summary of options is included below.

  -i, --input source  Run analysis on this audio file. Most uncompressed and
  compressed are supported, depending on how aubio was built.

  -o, --output sink  Save results in this file. The file will be created on
  the model of the input file. The detected frequency is played at the
  detected loudness.

  -r, --samplerate rate  Fetch the input source, resampled at the given
  sampling rate. The rate should be specified in Hertz as an integer. If 0,
  the sampling rate of the original source will be used. Defaults to 0.

  -B, --bufsize win  The size of the buffer to analyze, that is the length
  of the window used for spectral and temporal computations. Defaults to 2048.

  -H, --hopsize hop  The number of samples between two consecutive analysis.
  Defaults to 256.

  -p, --pitch method  The pitch detection method to use. See PITCH METHODS
  below. Defaults to 'default'.

  -u, --pitch-unit unit  The unit to be used to print frequencies. Possible
  values include midi, bin, cent, and Hz. Defaults to 'Hz'.

  -l, --pitch-tolerance thres  Set the tolerance for the pitch detection
  algorithm. Typical values range between 0.2 and 0.9. Pitch candidates found
  with a confidence less than this threshold will not be selected. The higher
  the threshold, the more confidence in the candidates. Defaults to unset.

  -s, --silence sil  Set the silence threshold, in dB, under which the onset
  will not be detected. A value of -20.0 would eliminate most onsets but the
  loudest ones. A value of -90.0 would select all onsets. Defaults to -90.0.

  -T, --timeformat format  Set time format (samples, ms, seconds). Defaults to
  seconds.

  -m, --mix-input  Mix source signal to the output signal before writing to
  sink.

  -f, --force-overwrite  Overwrite output file if it already exists.

  -j, --jack  Use Jack input/output. You will need a Jack connection
  controller to feed aubio some signal and listen to its output.

  -h, --help  Print a short help message and exit.
```

```
  -v, --verbose  Be verbose.

PITCH METHODS

  Available methods are:

  default  use the default method

  Currently, the default method is set to yinfft.

  schmitt  Schmitt trigger

  This pitch extraction method implements a Schmitt trigger to estimate the
  period of a signal. It is computationally very inexpensive, but also very
  sensitive to noise.

  fcomb  a fast harmonic comb filter

  This pitch extraction method implements a fast harmonic comb filter to
  determine the fundamental frequency of a harmonic sound.

  mcomb  multiple-comb filter

  This fundamental frequency estimation algorithm implements spectral
  flattening, multi-comb filtering and peak histogramming.

  specacf  Spectral auto-correlation function

  yin  YIN algorithm

  This algorithm was developed by A. de Cheveigne and H. Kawahara and
  was first published in:

  De Cheveigné, A., Kawahara, H. (2002) "YIN, a fundamental frequency
  estimator for speech and music", J. Acoust. Soc. Am. 111, 1917-1930.

  yinfft  Yinfft algorithm

  This algorithm was derived from the YIN algorithm. In this implementation, a
  Fourier transform is used to compute a tapered square difference function,
  which allows spectral weighting. Because the difference function is tapered,
  the selection of the period is simplified.

  Paul Brossier, Automatic annotation of musical audio for interactive systems,
  Chapter 3, Pitch Analysis, PhD thesis, Centre for Digital music, Queen Mary
  University of London, London, UK, 2006.

SEE ALSO

  aubioonset(1),
  aubiotrack(1),
  aubionotes(1),
  aubioquiet(1),
  aubiomfcc(1),
  and
  aubiocut(1).

AUTHOR
```

```
  This manual page was written by Paul Brossier <piem@aubio.org>. Permission is
  granted to copy, distribute and/or modify this document under the terms of
  the GNU General Public License as published by the Free Software Foundation,
  either version 3 of the License, or (at your option) any later version.
```

aubiomfcc

```
NAME
  aubiomfcc - a command line tool to compute Mel-Frequency Cepstrum Coefficients

SYNOPSIS

  aubiomfcc source
  aubiomfcc [[-i] source]
            [-r rate] [-B win] [-H hop]
            [-T time-format]
            [-v] [-h]

DESCRIPTION

  aubiomfcc compute the Mel-Frequency Cepstrum Coefficients (MFCC).

  MFCCs are coefficients that make up for the mel-frequency spectrum, a
  representation of the short-term power spectrum of a sound. By default, 13
  coefficients are computed using 40 filters.

  When started with an input source (-i/--input), the coefficients are given on
  the console, prefixed by their timestamps in seconds.

OPTIONS

  This program follows the usual GNU command line syntax, with long options
  starting with two dashes (--). A summary of options is included below.

  -i, --input source  Run analysis on this audio file. Most uncompressed and
  compressed are supported, depending on how aubio was built.

  -r, --samplerate rate  Fetch the input source, resampled at the given
  sampling rate. The rate should be specified in Hertz as an integer. If 0,
  the sampling rate of the original source will be used. Defaults to 0.

  -B, --bufsize win  The size of the buffer to analyze, that is the length
  of the window used for spectral and temporal computations. Defaults to 512.

  -H, --hopsize hop  The number of samples between two consecutive analysis.
  Defaults to 256.

  -T, --timeformat format  Set time format (samples, ms, seconds). Defaults to
  seconds.

  -h, --help  Print a short help message and exit.

  -v, --verbose  Be verbose.

REFERENCES
```

```
  Using the default parameters, the filter coefficients will be computed
  according to Malcolm Slaney's Auditory Toolbox, available at the following
  url:

  http://cobweb.ecn.purdue.edu/~malcolm/interval/1998-010/ (see file mfcc.m)

SEE ALSO

  aubioonset(1),
  aubiopitch(1),
  aubiotrack(1),
  aubionotes(1),
  aubioquiet(1),
  and
  aubiocut(1).

AUTHOR

  This manual page was written by Paul Brossier <piem@aubio.org>. Permission is
  granted to copy, distribute and/or modify this document under the terms of
  the GNU General Public License as published by the Free Software Foundation,
  either version 3 of the License, or (at your option) any later version.
```

aubiotrack

```
NAME
  aubiotrack - a command line tool to extract musical beats from audio signals

SYNOPSIS

  aubiotrack source
  aubiotrack [[-i] source] [-o sink]
             [-r rate] [-B win] [-H hop]
             [-T time-format]
             [-s sil] [-m]
             [-j] [-N miditap-note] [-V miditap-velo]
             [-v] [-h]

DESCRIPTION

  aubiotrack attempts to detect beats, the time where one would intuitively be
  tapping his foot.

  When started with an input source (-i/--input), the detected beats are given
  on the console, in seconds.

  When started without an input source, or with the jack option (-j/--jack),
  aubiotrack starts in jack mode.

OPTIONS

  This program follows the usual GNU command line syntax, with long options
  starting with two dashes (--). A summary of options is included below.

  -i, --input source  Run analysis on this audio file. Most uncompressed and
```

```
  compressed are supported, depending on how aubio was built.

  -o, --output sink  Save results in this file. The file will be created on the
  model of the input file. Beats are marked by a short wood-block like sound.

  -r, --samplerate rate  Fetch the input source, resampled at the given
  sampling rate. The rate should be specified in Hertz as an integer. If 0,
  the sampling rate of the original source will be used. Defaults to 0.

  -B, --bufsize win  The size of the buffer to analyze, that is the length
  of the window used for spectral and temporal computations. Defaults to 512.

  -H, --hopsize hop  The number of samples between two consecutive analysis.
  Defaults to 256.

  -s, --silence sil  Set the silence threshold, in dB, under which the pitch
  will not be detected. A value of -20.0 would eliminate most onsets but the
  loudest ones. A value of -90.0 would select all onsets. Defaults to -90.0.

  -m, --mix-input  Mix source signal to the output signal before writing to
  sink.

  -f, --force-overwrite  Overwrite output file if it already exists.

  -j, --jack  Use Jack input/output. You will need a Jack connection
  controller to feed aubio some signal and listen to its output.

  -N, --miditap-note  Override note value for MIDI tap. Defaults to 69.

  -V, --miditap-velop  Override velocity value for MIDI tap. Defaults to 65.

  -T, --timeformat format  Set time format (samples, ms, seconds). Defaults to
  seconds.

  -h, --help  Print a short help message and exit.

  -v, --verbose  Be verbose.

BEAT TRACKING METHODS

  Aubio currently implements one the causal beat tracking algorithm designed by
  Matthew Davies and described in the following articles:

  Matthew E. P. Davies and Mark D. Plumbley. Causal tempo tracking of audio.
  In Proceedings of the International Symposium on Music Information Retrieval
  (ISMIR), pages 164169, Barcelona, Spain, 2004.

  Matthew E. P. Davies, Paul Brossier, and Mark D. Plumbley. Beat tracking
  towards automatic musical accompaniment. In Proceedings of the Audio
  Engineering Society 118th Convention, Barcelona, Spain, May 2005.

SEE ALSO

  aubioonset(1),
  aubiopitch(1),
  aubionotes(1),
  aubioquiet(1),
  aubiomfcc(1),
```

```
  and
  aubiocut(1).

AUTHOR

  This manual page was written by Paul Brossier <piem@aubio.org>. Permission is
  granted to copy, distribute and/or modify this document under the terms of
  the GNU General Public License as published by the Free Software Foundation,
  either version 3 of the License, or (at your option) any later version.
```

aubionotes

```
NAME
  aubionotes - a command line tool to extract musical notes

SYNOPSIS

  aubionotes source
  aubionotes [[-i] source]
             [-r rate] [-B win] [-H hop]
             [-O method] [-t thres]
             [-p method] [-u unit] [-l thres]
             [-T time-format]
             [-s sil]
             [-j] [-v] [-h]

DESCRIPTION

  aubionotes attempts to detect notes by looking for note onsets and pitches.
  Consecutive events are segmented using onset detection, while a fundamental
  frequency extraction algorithm determines their pitch.

  When started with an input source (-i/--input), the detected notes are
  printed on standard output, in seconds and midi note number.

  When started without an input source, or with the jack option (-j/--jack),
  aubionotes starts in jack mode.

OPTIONS

  This program follows the usual GNU command line syntax, with long options
  starting with two dashes (--). A summary of options is included below.

  -i, --input source  Run analysis on this audio file. Most uncompressed and
  compressed are supported, depending on how aubio was built.

  -r, --samplerate rate  Fetch the input source, resampled at the given
  sampling rate. The rate should be specified in Hertz as an integer. If 0,
  the sampling rate of the original source will be used. Defaults to 0.

  -B, --bufsize win  The size of the buffer to analyze, that is the length
  of the window used for spectral and temporal computations. Defaults to 512.

  -H, --hopsize hop  The number of samples between two consecutive analysis.
  Defaults to 256.
```

```
  -O, --onset method  The onset detection method to use. See ONSET METHODS
  below. Defaults to 'default'.

  -t, --onset-threshold thres  Set the threshold value for the onset peak
  picking. Typical values are typically within 0.001 and 0.900. Defaults to
  0.1. Lower threshold values imply more onsets detected. Try 0.5 in case of
  over-detections. Defaults to 0.3.

  -M, --minioi value  Set the minimum inter-onset interval, in seconds, the
  shortest interval between two consecutive notes. Defaults to 0.030

  -p, --pitch method  The pitch detection method to use. See PITCH METHODS
  below. Defaults to 'default'.

  -u, --pitch-unit unit  The unit to be used to print frequencies. Possible
  values include midi, bin, cent, and Hz. Defaults to 'Hz'.

  -l, --pitch-tolerance thres  Set the tolerance for the pitch detection
  algorithm. Typical values range between 0.2 and 0.9. Pitch candidates found
  with a confidence less than this threshold will not be selected. The higher
  the threshold, the more confidence in the candidates. Defaults to unset.

  -s, --silence sil  Set the silence threshold, in dB, under which the pitch
  will not be detected. A value of -20.0 would eliminate most onsets but the
  loudest ones. A value of -90.0 would select all onsets. Defaults to -90.0.

  -T, --timeformat format  Set time format (samples, ms, seconds). Defaults to
  seconds.

  -j, --jack  Use Jack input/output. You will need a Jack connection
  controller to feed aubio some signal and listen to its output.

  -h, --help  Print a short help message and exit.

  -v, --verbose  Be verbose.

ONSET METHODS

  Available methods: default, energy, hfc, complex, phase, specdiff, kl, mkl,
  specflux.

  See aubioonset(1) for details about these methods.

PITCH METHODS

  Available methods: default, schmitt, fcomb, mcomb, specacf, yin, yinfft.

  See aubiopitch(1) for details about these methods.

SEE ALSO

  aubioonset(1),
  aubiopitch(1),
  aubiotrack(1),
  aubioquiet(1),
  aubiomfcc(1),
  and
  aubiocut(1).
```

```
AUTHOR

  This manual page was written by Paul Brossier <piem@aubio.org>. Permission is
  granted to copy, distribute and/or modify this document under the terms of
  the GNU General Public License as published by the Free Software Foundation,
  either version 3 of the License, or (at your option) any later version.
```

aubioquiet

```
NAME
  aubioquiet - a command line tool to extracts quiet and loud regions from a file

SYNOPSIS

  aubioquiet source
  aubioquiet [[-i] source]
             [-r rate] [-B win] [-H hop]
             [-T time-format]
             [-s sil]
             [-v] [-h]

DESCRIPTION

  aubioquiet will print a timestamp each time it detects a new silent region or
  a new loud region in a sound file.

  When started with an input source (-i/--input), the detected timestamps are
  printed on the console, in seconds.

OPTIONS

  This program follows the usual GNU command line syntax, with long options
  starting with two dashes (--). A summary of options is included below.

  -i, --input source  Run analysis on this audio file. Most uncompressed and
  compressed are supported, depending on how aubio was built.

  -r, --samplerate rate  Fetch the input source, resampled at the given
  sampling rate. The rate should be specified in Hertz as an integer. If 0,
  the sampling rate of the original source will be used. Defaults to 0.

  -B, --bufsize win  The size of the buffer to analyze, that is the length
  of the window used for spectral and temporal computations. Defaults to 512.

  -H, --hopsize hop  The number of samples between two consecutive analysis.
  Defaults to 256.

  -s, --silence sil  Set the silence threshold, in dB, under which the pitch
  will not be detected. Defaults to -90.0.

  -T, --timeformat format  Set time format (samples, ms, seconds). Defaults to
  seconds.

  -h, --help  Print a short help message and exit.
```

```
  -v, --verbose  Be verbose.

EXAMPLE OUTPUT

  NOISY: 28.775330

  QUIET: 28.914648


SEE ALSO

  aubioonset(1),
  aubiopitch(1),
  aubiotrack(1),
  aubionotes(1),
  aubiomfcc(1),
  and
  aubiocut(1).

AUTHOR

  This manual page was written by Paul Brossier <piem@aubio.org>. Permission
  is granted to copy, distribute and/or modify this document under the terms
  of the GNU General Public License, Version 3 any later version published
  by the Free Software Foundation.
```

Command line features

feat vs. prg	onset	pitch	mfcc	track	notes	quiet	cut1	short options
input	Y	Y	Y	Y	Y	Y	Y	-i
output	Y	Y	N	Y	Y	N	Y!1	-o,-m,-f
Hz/buf/hop	Y	Y	Y	Y	Y	Y!2	Y	-r,-B-,H
jack	Y	Y	N	Y	Y	N!3	N	-j
onset	Y	N	N	Y!8	Y!6	N	Y	-O,-t,-M
pitch	N	Y	N	N	Y!6	N	N!5	-p,-u,-l
silence	Y	Y	N	Y	Y!7	Y	N!4	-s
timefmt	Y	Y	Y	Y	Y	Y	!	-T
help	Y	Y	Y	Y	Y	Y	Y	-h
verbose	Y	Y	Y	Y	Y	Y	Y	-v

1. `aubiocut --output` is used to specify a directory, not a file.
2. Option `--bufsize` is useless for `aubioquiet`
3. `aubioquiet` could have a jack output
4. Regression, re-add slicing at silences to `aubiocut`
5. `aubiocut` could cut on notes
6. `aubionotes` needs onset/pitch setters.
7. Silence was different for pitch and onset, test.
8. Some `aubiotrack` options should be disabled (minioi, threshold).

Developping with aubio

Here is a brief overview of the C library.

For a more detailed list of available functions, see the API documentation.

To report issues, ask questions, and request new features, use Github Issues

Design Basics

The library is written in C and is optimised for speed and portability.

All memory allocations take place in the *new_* methods. Each successful call to *new_* should have a matching call to *del_* to deallocate the object.

```
// new_ to create an object foobar
aubio_foobar_t * new_aubio_foobar(void * args);
// del_ to delete foobar
void del_aubio_foobar (aubio_foobar_t * foobar);
```

The main computations are done in the *_do* methods.

```
// _do to process output = foobar(input)
audio_foobar_do (aubio_foobar_t * foobar, fvec_t * input, cvec_t * output);
```

Most parameters can be read and written at any time:

```
// _get_param to get foobar.param
smpl_t aubio_foobar_get_a_parameter (aubio_foobar_t * foobar);
// _set_param to set foobar.param
uint_t aubio_foobar_set_a_parameter (aubio_foobar_t * foobar, smpl_t a_parameter);
```

In some case, more functions are available:

```
// non-real time functions
uint_t aubio_foobar_reset(aubio_foobar_t * t);
```

Basic Types

```
// integers
uint_t n = 10;                  // unsigned
sint_t delay = -90;             // signed

// float
smpl_t a = -90.;                // simple precision
lsmp_t f = 0.024;               // double precision

// vector of floats (simple precision)
fvec_t * vec = new_fvec(n);
vec->data[0] = 1;
vec->data[vec->length-1] = 1.; // vec->data has n elements
fvec_print(vec);
del_fvec(vec);

// complex data
```

```
cvec_t * fftgrain = new_cvec(n);
vec->norm[0] = 1.;              // vec->norm has n/2+1 elements
vec->phas[n/2] = 3.1415;        // vec->phas as well
del_cvec(fftgrain);

// matrix
fmat_t * mat = new_fmat (height, length);
mat->data[height-1][0] = 1;     // mat->data has height rows
mat->data[0][length-1] = 10;    // mat->data[0] has length columns
del_fmat(mat);
```

Reading a sound file

In this example, aubio_source is used to read a media file.

First, define a few variables and allocate some memory.

```
  uint_t samplerate = 0;
  uint_t hop_size = 256;
  uint_t n_frames = 0, read = 0;

  aubio_source_t* s =
    new_aubio_source(source_path, samplerate, hop_size);
  fvec_t *vec = new_fvec(hop_size);
```

Note: With `samplerate = 0`, `aubio_source` will be created with the file's original samplerate.

Now for the processing loop:

```
  do {
    aubio_source_do(s, vec, &read);
    fvec_print (vec);
    n_frames += read;
  } while ( read == hop_size );
```

At the end of the processing loop, memory is deallocated:

```
  del_fvec (vec);
  del_aubio_source (s);
```

See the complete example: `test-source.c`.

Computing a spectrum

Now let's create a phase vocoder:

```
  uint_t win_s = 32; // window size
  uint_t hop_s = win_s / 4; // hop size

  fvec_t * in = new_fvec (hop_s); // input buffer
  cvec_t * fftgrain = new_cvec (win_s); // fft norm and phase
  fvec_t * out = new_fvec (hop_s); // output buffer
```

The processing loop could now look like:

```
while ( n-- ) {
  // get some fresh input data
  // ..

  // execute phase vocoder
  aubio_pvoc_do (pv,in,fftgrain);

  // do something with fftgrain
  // ...
  cvec_print (fftgrain);

  // optionally rebuild the signal
  aubio_pvoc_rdo(pv,fftgrain,out);

  // and do something with the result
  // ...
  fvec_print (out);
}
```

Time to clean up the previously allocated memory:

```
// clean up
del_fvec(in);
del_cvec(fftgrain);
del_fvec(out);
del_aubio_pvoc(pv);
aubio_cleanup();
```

See the complete example: `test-phasevoc.c`.

Doxygen documentation

The latest version of the API documentation is built using Doxygen and is available at:

https://aubio.org/doc/latest/

Contribute

Please report any issue and feature request at the Github issue tracker. Patches and pull-requests welcome!

About

This library gathers a collection of music signal processing algorithms written by several people. The documentation of each algorithms contains a brief description and references to the corresponding papers.

Credits

Many thanks to everyone who contributed to aubio, including:

- Martin Hermant (MartinHN)
- Eduard Müller (emuell)

- Nils Philippsen (nphilipp)
- Tres Seaver (tseaver)
- Dirkjan Rijnders (dirkjankrijnders)
- Jeffrey Kern (anwserman)
- Sam Alexander (sxalexander)

Special thanks to Juan Pablo Bello, Chris Duxbury, Samer Abdallah, Alain de Cheveigne for their help. Also many thanks to Miguel Ramirez and Nicolas Wack for their advices and help fixing bugs.

Publications

Substantial informations about several of the algorithms and their evaluation are gathered in:

- Paul Brossier, Automatic annotation of musical audio for interactive systems, PhD thesis, Centre for Digital music, Queen Mary University of London, London, UK, 2006.

Additional results obtained with this software were discussed in the following papers:

- P. M. Brossier and J. P. Bello and M. D. Plumbley, Real-time temporal segmentation of note objects in music signals in *Proceedings of the International Computer Music Conference*, 2004, Miami, Florida, ICMA
- P. M. Brossier and J. P. Bello and M. D. Plumbley, *Fast labelling of note objects in music signals <https://aubio.org/articles/brossier04fastnotes.pdf>*, in *Proceedings of the International Symposium on Music Information Retrieval*, 2004, Barcelona, Spain

Citation

Please refer to the Zenodo link in the file README.md to cite this release.

Copyright

License