CleanTiPy

Release 0.6

Raphaël LEIBA

Jan 07, 2025

Contents

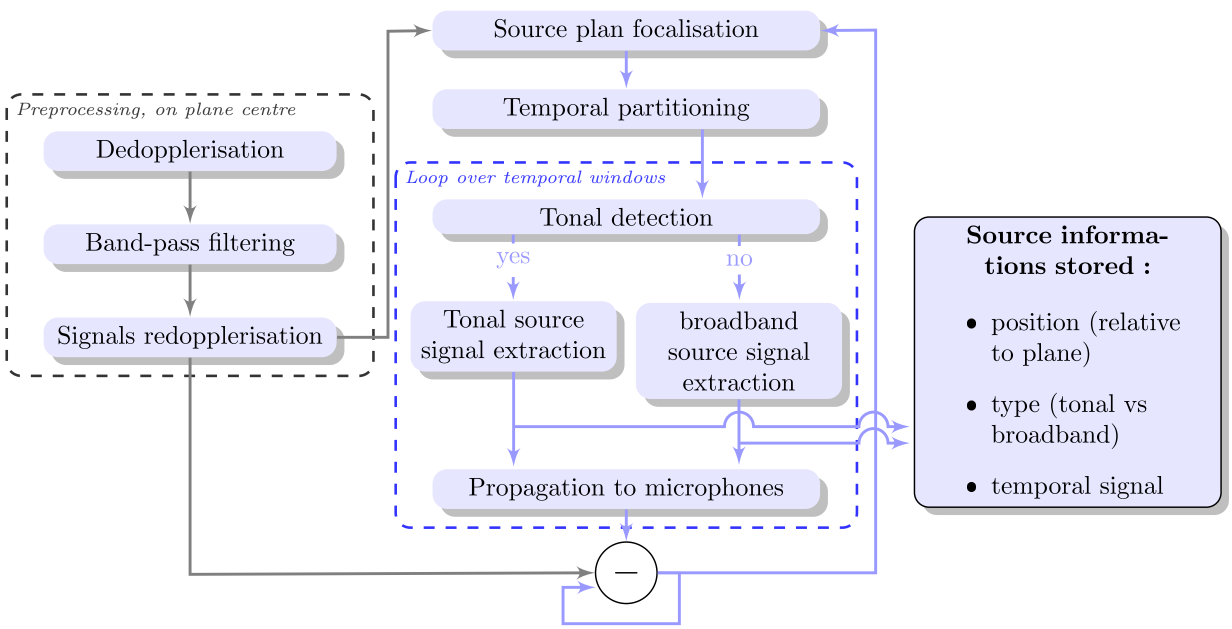
1 Principle of the method

The schematics bellow presents the CLEAN-T algorithm implemented in this repository.

> **Note**
>
> The subject is supposed in this example to be a plane but it can be any object as long as you provide its trajectory and a possible source grid sourounding it.

As depicted in the central part of the schematics, CLEAN-T is based on the used of the time-domain formulation of moving source propagation and beaforming to iteratively :

- compute the beamforming over the given grid (or focal plan)
- localise the maximum
- isolate the temporal signal comming from the maximum location
- propagate the signal to the microphones
- substract the propagated signals to the previously stored signals



If a multi-frequency analysis is performed, the left block of the schematics is performed in order to filter each microphone signal in the moving-source related domain (thus dedopplerising, filtering and re-dopplering the signals).

2 Usage

2.1 Requirement

To use CleanTiPy, first install the required packages :

- numpy
- matplotlib
- scipy
- pyfftw (optional, to speed up the discrete Fourier transforms (DFT) in `DeconvolutionMethods.CleanT.find_max()`).
- simplespectral (uses pyfftw, scipy.fft or numpy.fft seamlessly)
- joblib

> **Note**
>
> pyfftw requires FFTW3 to function. FFTW3 is available under two licenses, the free GPL and a non-free license that allows it to be used in proprietary program

2.2 Installation

This code is developed in Python 3.11 and therefore back-compatibility is not guaranteed.

Install the required packages with

```
pip install -r requirements.txt
```

2.3 Examples

For a multi-frequency analysis you can run this example:

```
cd ./Examples/
python computeCleanT_multiFreq.py
```

For a multi-frequency analysis with different angular selection windows you can run this example:

```
cd ./Examples/
python computeCleanT_multiFreq_multiAngles.py
```

3 Structure

3.1 Overall structure

The project is based on the use of two main classes:

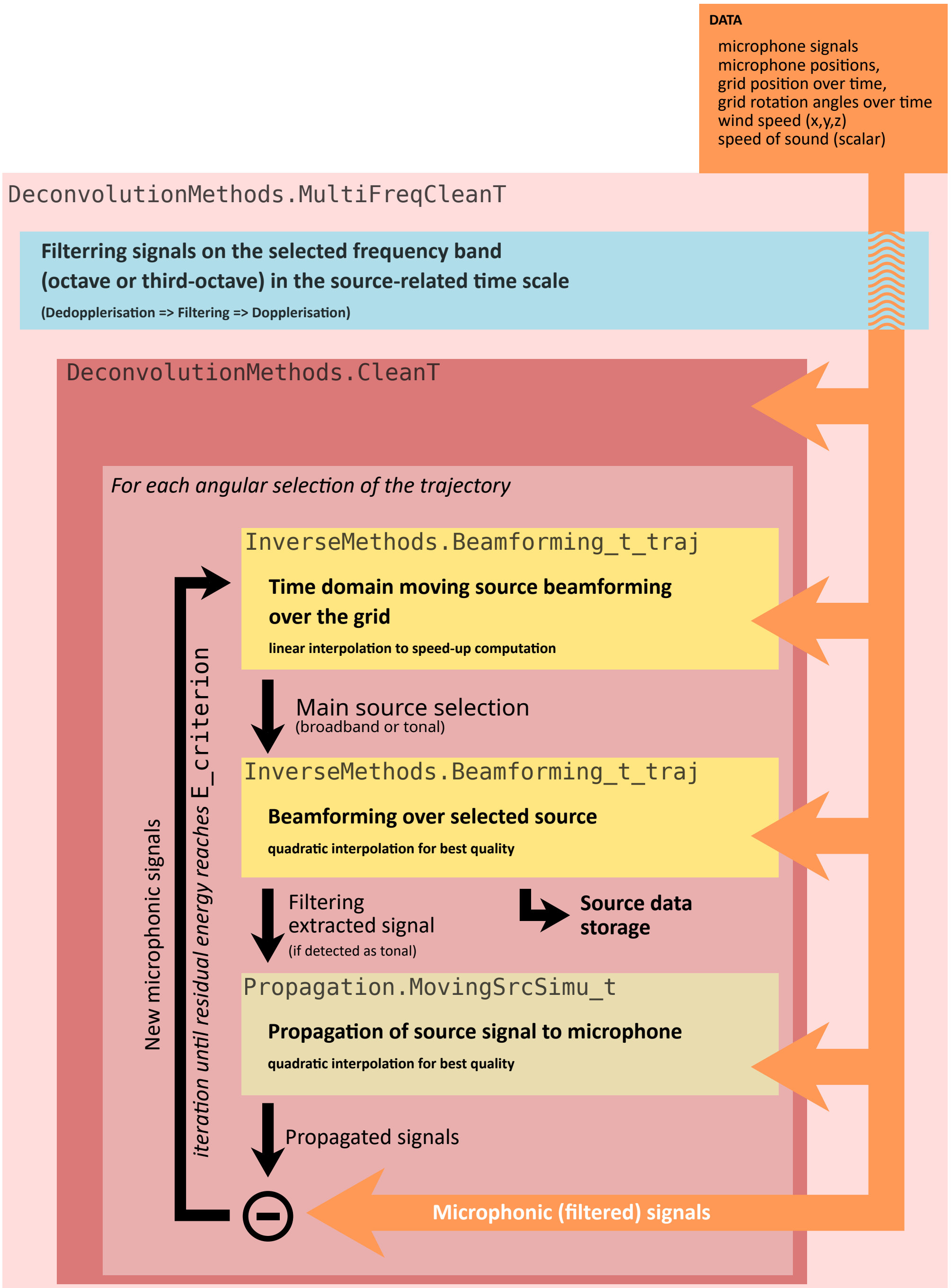
- the `Propagation.MovingSrcSimu_t` class that computes the propagation of the sound from a moving source.
- the `InverseMethods.Beamforming_t_traj` class that computes the beamforming over a moving source in the time domain.

They are both used to build the CLEAN-T class :

- the `DeconvolutionMethods.CleanT` class computes CLEAN-T algorithm for a moving source.
- the `DeconvolutionMethods.MultiFreqCleanT` class is a wrap of `DeconvolutionMethods.CleanT` to separate the analysis by frequency band (octave or third-octave bands)

All sources are assumed to be monopolar and the sound propagation occurs in a homogeneous medium with homogeneous wind.

The class links to implement the CLEAN-T method are detailed in the figure below:



In details, CLEAN-T class is initialised with theses parameters:

`DeconvolutionMethods.CleanT.__init__`(*self*, *geom*, *grid*, *traj*, *t*, *Sig*, *angles=None*, *t_traj=None*, *N_iter=50*, *alpha=1*, *E_criterion=0.05*, *E_convergence_criterion=0.5*, *angleSelection=None*, *theta=None*, *debug=False*, *fc=None*, *c=343*, *Mach_w=None*, *findTonal=True*, *monitor=False*)

Parameters

geom
: [numpy array] geometry of the array (Nmic x 3)

grid
: [numpy array] grid points (Ni x 3)

traj
: [numpy array] trajectory of the grid (Nt_traj x 3)

t
: [numpy array] time vector of the microphone signals (Nt)

p_t
: [numpy array] microphone signals (Nmic x Nt)

angles
: [numpy array] rotations around x,y and z in rad (Nt x 3)

t_traj
: [numpy array] time vector for the trajectory (Nt_traj)

N_iter
: [int] maximum number of iteration

alpha
: [float] proportion of source amplitude to be removed at each iteration

E_criterion
: [float] Stop criterion base on the ratio of remaining energy (default: 0.05 for 5%)

E_convergence_criterion
: [float] Stop criterion base on the convergence of remaining energy (default: 0.5%)

  If the reduction in energy between two iteration is less than *E_convergence_criterion* execution stops

angleSelection
: [numpy array] Limits of angle windows in degree (Nwin x 2). The CLEAN-T process will be performed for each angle window

debug
: [boolean] if true displays debugging prints

fc
: [float (optional)] central frequency of analysis (interesting for monitoring, not needed otherwise)

c
: [float, scalar] Speed of sound

Mach_w
: [numpy array (3,)] Mach number of the wind along x, y, z

findTonal
: [boolean] default True, il false, dont look for tonal source and only process sources broadband

monitor
: [boolean] if true displays monitoring prints and graphs

3.2 File structure

The project is structured over three modules :

- the `Propagation` module that hold the classes for the direct path : the propagation of the sound.
- the `InverseMethods` module that hold the classes for the inverse methods : the back-propagation of the sound.
- the `DeconvolutionMethods` module that hold the classes for CLEAN-T algorithm.

Propagation

Library generating simulated data

Created on Fri Apr 1 15:41:51 2022 @author: rleiba

class `Propagation.`**`MovingSrcSimu_t`**(*geom*, *pos*, *traj*, *t*, *sig*, *t_traj=None*, *angles=None*, *SNR=None*, *timeOrigin='source'*, *debug=False*, *c=343*, *Mach_w=None*)

Class to compute the simulated signal received by the array microphones for a static source. Computation in time domain.

class `Propagation.`**`StaticSrcSimu_t`**(*geom*, *pos*, *t*, *sig*, *SNR=None*)

Class to compute the simulated signal received by the array microphones for a static source. Computation in time domain.

InverseMethods

Library computing different aspects of beamforming, PSF or other tools

Created on Fri Apr 1 14:33:53 2022 @author: rleiba

class `InverseMethods.`**`Beamforming_f`**(*geom*, *grid*, *f*, *p_f*, *isMicActive*)

Class to compute the classical frequency domain beamforming

class `InverseMethods.`**`Beamforming_t`**(*geom*, *grid*, *t*, *p_t*)

Class to compute temporal domain beamforming

class `InverseMethods.`**`Beamforming_t_traj`**(*geom*, *grid*, *traj*, *t*, *p_t*, *angles=None*, *t_traj=None*, *debug=False*, *QuantitativeComputation=True*, *internVar_dtype=<class 'numpy.float64'>*, *c=343*, *Mach_w=None*)

Class to compute temporal domain beamforming over a trajectory

`compute`(*parrallel=True*, *interpolation='linear'*)

Function computing the beamforming

Parameters

parrallel

[bool] Parrallelize the computation (default=True)

interpolation

[str] select the interpolation type from scipy.interpolate.interp1d : 'linear', 'nearest', 'nearest-up', 'zero', 'slinear', 'quadratic', 'cubic', 'previous', or 'next'.

'quadratic' recomended for better results and 'linear' for good compromize (computation time vs quality) class InverseMethods.**PointSpreadFunction**(*geom*, *grid*, *f*, *isMicActive*)

Class to compute the Point Spread Fucntion of a microphone array based on its geometry, the grid and the source position in the grid plane

InverseMethods.**frequencyBand**(*data*, *f*, *fc=[1000]*, *type='octave'*)

Compute SPL from data on octave bands

Parameters

data
[numpy array] data in shape (freq, n_data)

f
[numpy array] frequency axis

fc
[list] Center frequencies of octave bands to process (Hz).

type
[string] chose between 'octave' for octave bands or 'third' for third-octave bands

DeconvolutionMethods

CLEAN-T core functions

Created on Mon Jun 19 2023 @author: rleiba

class DeconvolutionMethods.**CleanT**(*geom*, *grid*, *traj*, *t*, *Sig*, *angles=None*, *t_traj=None*, *N_iter=50*, *alpha=1*, *E_criterion=0.05*, *E_convergence_criterion=0.5*, *angleSelection=None*, *theta=None*, *debug=False*, *fc=None*, *c=343*, *Mach_w=None*, *findTonal=True*, *monitor=False*)

Class to compute CLEAN-T, time-domain variant of the CLEAN deconvolution algorithm

computeAngleWindows()

compute self.TemporalMask: a smooth mask in the interpolated (microphone signals sample frequency) trajectory time scale for each angular window

computeAngleWindows_traj()

compute self.TrajMask: a binary mask in the trajectory time scale for each angular window:

0: trajectory point not in angular selection

1: trajectory point in angular selection

find_max(*dB_*, *aa=None*, *parrallel=True*)

find_max take the beamformed signals over the grid in order to localise the main source of the grid

Returns

i_max: int
index of the source (assumed to be the most energetic bin of the beamforming)

src_type: int
type of source: 0 (broadband) or 1 (tonal)

set_envs()

Sets the propagation and beaforming environements that will be used during the iterative process

class DeconvolutionMethods.**MultiFreqCleanT**(*geom*, *grid*, *traj*, *t*, *Sig*, *angles=None*, *t_traj=None*, *N_iter=50*, *alpha=1*, *E_criterion=0.05*, *E_convergence_criterion=0.5*, *debug=False*, *fc=[500, 1000]*, *bandtype='octave'*, *isMicActive=None*, *angleSelection=None*, *c=343*, *Mach_w=None*, *findTonal=True*, *monitor=False*)

Class to compute CLEAN-T, time-domain variant of the CLEAN deconvolution algorithm

ComputeCleanT(*dyn=30*, *parrallel=False*)

Compute CLEAN-T for each frequency band

Python Module Index

d

i

p