Overview

As Python has become the go-to language for machine learning, having a package to streamline the prototyping of microphone array algorithms and for accurate room impulse response (RIR) generation is important. We present **Pyroomacoustics**. Three key features include:

**Object-oriented interface**

- Convenietly construct different simulation scenarios in 2D/3D.

**Room impulse response generator**

- C implementation of the image source model to generate RIRs and simulate propagation between sources and microphones.

**Reference algorithms**

- Beamforming, direction-of-arrival (DOA), adaptive filtering, Short-Time Fourier Transform (STFT).

And it's free!

![Image of microphone array](https://example.com/microphone_array.png)

**Constructing a scene**

With Pyroomacoustics, we can build a 2D or 3D room in a very intuitive fashion and conveniently add a source and microphone array.

- Impact Energy as ep
- Impulse response as pt
- Room shape as e.g. cuboid
- Impact Pyroomacoustics as pro

**Specify signal source**

- Fs, Signal as well as read (READPATH)

- Create a 3D scene from the specified coordinates

- Create an array of the specified source coordinates

- Create the 3D scene by extending the 3D shape

- Add source and set the signal to WAV file content

- Add microphone array

- M = np.array([[0, 2, 4], [1, 2, 3]])

- [X, Y, Z] = np.meshgrid(x, y, z)

- Moreover, we can plot the constructed room using standard plotting libraries:

- num_plt()

- [x0, y0, z0] = plot_customization on current figure as = pt_grid() pt_shown()

Room impulse response (RIR) generator

The RIR generator is based on the image source model (ISM).

The impulse response between a source and an array depends on the distance between the source and the array. The impulse response between a source sampled at $t_n$ is given by:

$$h[m,n,t] = \left(1 - \frac{1}{\lambda} (m-1) \pi \right) \delta[m-1] \delta[n-1] \delta(t-nT)$$

where $\lambda$ is the wavelength of the sound, $m$ is the number of source, and $n$ is the number of sample. The sampling frequency, maximum ISM order and absorption factor can be set when constructing the room.

**The RIRs**

- The RIRs can be computed and plotted with the following commands:

- num.plot_rir():

- num.plot_rir(order=12, absorption=0.15)

To simulate the recording of the source with the specified microphone array:

- num.compute_rir():

- num.process():

Most available software can only compute the RIR for shoebox rooms. With pyroomacoustics, it is possible to compute the RIR for arbitrary polyhedral rooms!

Beamforming (time and frequency)

Classic beamforming algorithms (DAS and MVDR) are included as special cases of the acoustic rake receivers.

- Create a sound source:

- [X, Y, Z] = np.meshgrid(x, y, z)

- [x0, y0, z0] = plot_customization on current figure as = pt_grid() pt_shown()

- Shoeebox room (4x8) with one source (circle) and one interferer (square) and 6 micr circular (3.5 mm radius) array with a center mic.

The following command can be used to beamform the simulated microphone signals:

- num.compute_rake():

- num.process():

- adaptation_alg=

Direction-of-arrival (DOA)

**DOA syntax where num: one of [SRP, MUSIC, CSSM, WAVES, FRIDA].**

- doa = num.compute_doa(num.geometry, num.sample, num.process, num.lapack)

- num.compute_doa(num.geometry, num.sample, num.process, num.lapack)

Spatial spectrum stored in magnitude and estimated directions in magnitude and maximum for 3D.

- (Right) far-field DOA with the same geometry as in "Beamforming." Signal consists of two 1-second long white noise sources at 61° and 270° with an SNR of 0 dB. DOA is performed within the frequency range of [300, 3500] Hz, i.e. range of human speech.

Adaptive filtering

**Adaptive filt. syntax where num: one of [NLMS, BlockLMS, RLS, BlockRLS].**

- f_obj = num.compute_filter(num.geometry, num.sample, num.lapack)

- f_obj = num.compute_filter(num.geometry, num.sample, num.lapack)

- Estimated filter fi stored in magnitude.

- (Right) Converged different adaptive filtering methods for an SNR of 15 dB.

**STFT engine and real-time processing**

While there is an STFT module supporting overlap-add, zero-padding, and various analysis/synthesis windows:

- 

It performs the analysis and synthesis operations on the entire signal. The num: real-time STFT class is more suitable for streaming / real-time data and is applicable to multi-channel.

- 

**Related publications**


Check the paper for references to all algorithms implemented in pyroomacoustics!

---

May the Fork be with you!

New features on the horizon! If you would like to make a contribution, feel free to make a pull request by navigating to the link below:

https://github.com/lsys/pyroomacoustics