sound_source_id

Release 0.2.8

Samuele Carcagno

Jul 26, 2024

CONTENTS:

CHAPTER

ONE

SOUND_SOURCE_ID

sound_source_id is a program for testing sound localization. The interface is shown in Figure *Screenshot of the sound_source_id interface*.

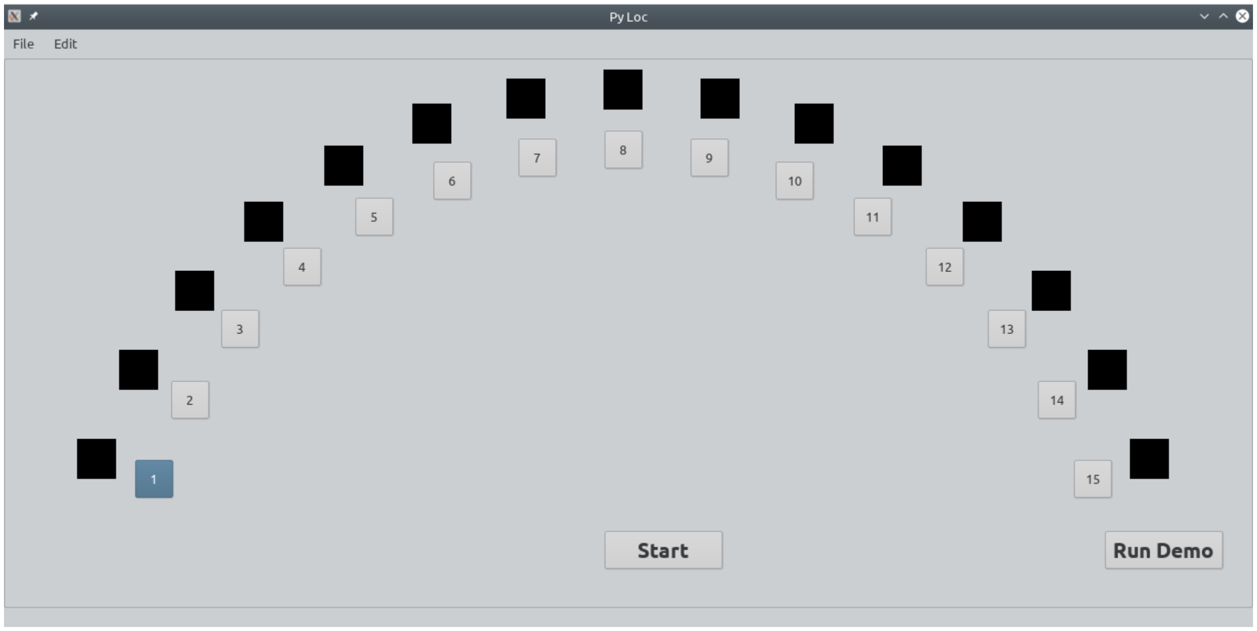


Fig. 1: Screenshot of the sound_source_id interface
sound_source_id supports presenting sounds through a physical array of speakers laid out in a circular (or spherical, if multiple elevations are used) layout, or through earphones. In the latter case, spazialization is achieved by convolving the stimuli with an head-related transfer function (which must be provided by the used through a SOFA file).

CHAPTER

TWO

INSTALLATION

sound_source_id has been successfully installed and used on Linux and Windows. It should also work on Mac platforms, but this has not been tested. sound_source_id is written in Python and can be installed via pip:

```
pip install sound_source_id
```

sound_source_id depends on a few Python modules including:

- PyQt6
- numpy
- scipy
- matplotlib
- pandas
- PyAudio https://pypi.org/project/PyAudio/

depending on your Python distribution you may want to install these dependecies before installing sound_source_id via pip (e.g. through conda if you're using the Anaconda Python distribution or through your Linux distribution package manager if you're using the Python installation that comes with your Linux distribution), otherwise pip will attempt to automatically pull in and install these dependencies. If the program is successfully installed you should be able to start it from a bash/DOS terminal with the command:

```
sound_source_id
```

You need to ensure that the Python environment you're using when you call the above command matches the one you used when you installed the application.

Sound can be played with either PyAudio, or SoX on Windows. On Linux pyalsaaudio can be also used. Depending on how you want sound to be played, you need to install:

- pyalsaaudio https://pypi.org/project/pyalsaaudio/

or SoX:

- https://sox.sourceforge.net/

CHAPTER

THREE

PARAMETERS FILE

The settings for a test are stored in a parameters file, which is a plain text file. An example parameters file for a setup with physical speakers is shown below:

```
mode = speakers
azimuths = -70, -60, -50, -40, -30, -20, -10, 0, 10, 20, 30, 40, 50, 60, 70
elevs = 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0
labels = 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15
channels = 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15
n_chan = 15
n_blocks = 1
stim_list_file = stim_list.csv
randomize = true
demo_stim = pink_noises/noise1.wav
demo_stim_lev = 65
```

this is what each field indicates:

- *mode*: [`speakers`, `earphones`], whether spatialization will be achieved by presenting the stimuli through a physical array of speakers, or virtually through earphones
- *azimuths*: the azimuth angles (in degrees) at which the sounds are presented. Note that a 0° angle indicates straight ahead, a 90° angle is to the right, and a -90° angle to the left
- *elevs*: the elevation angles (in degrees) at which the sounds are presented; (0°: median plane front, 90°: up, 180°: median plane back, 270: down°). This field can be omitted if all of the elevation angles are at 0°
- *labels*: a label for each of the angles, this can be a number or a letter (e.g., a, b, c, etc…)
- *channels*: the channel of the soundcard that will be used to present a sound at the corresponding azimuth/elevation coordinate
- *n_chan*: the total number of channels for the setup
- *n_blocks*: the number of blocks, that is, how many times the test will be repeated
- *stim_list_file*: the path (absolute or relative) to the file containing the stimulation list (see below)
- *randomize*: if *true* the stim_list_file will be shuffled before the repetition of each block
- *demo_stim*: the path to the WAV file to be used for the demo
- *demo_stim_lev*: the sound level (in dB SPL) to be used for the demo

Below is an example parameter file for a setup with virtual spatial presentation through earphones:

```
mode = earphones
azimuths = -70, -60, -50, -40, -30, -20, -10, 0, 10, 20, 30, 40, 50, 60, 70
labels = 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15
n_blocks = 1
stim_list_file = stim_list.csv
randomize = true
demo_stim = pink_noises/noise1.wav
demo_stim_lev = 65
sofa_file_path = /media/ntfsShared/archives/auditory/Materials/KU100_Far_Field/HRIR_
↪CIRC360.sofa
sofa_az_coords = anticlockwise
```

many of the fields in the parameter file above are the same as for a parameter file for a setup with physical speakers. The following additional fields are needed in a parameters file for virtual spatial presentation through earphones:

- *sofa_file_path*: the path (absolute or relative) to the SOFA file containing the HRTF to be used for spatialization
- *sofa_az_coords*: [`anticlockwise`, `clockwise`], whether the azimuth coordinates of the SOFA file are specified in a clockwise(0: front, 90: right, 180: back; 270: left), or anti-clockwise (0: front, 90: left, 180: back; 270: right) arrangement

In a parameter file for a setup with virtual spatial presentation through earphones the *channels* and *n_chan* fields are not needed, and will be ignored if present.

Some additional optional fields can be used to adjust the visual appearance of the widgets layout:

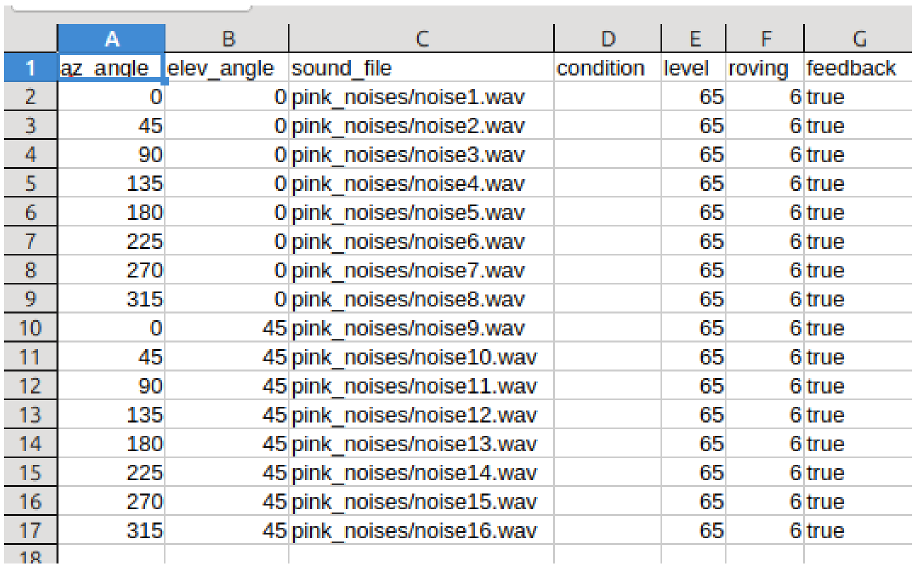
- *resp_bt_wd*: width of the response buttons in pixels (default 40)
- *resp_bt_ht*: height of the response buttons in pixels (default 40)
- *play_bt_wd*: width of the play buttons in pixels (default 40)
- *play_bt_ht*: height of the play buttons in pixels (default 40)
- *resp_lt_wd*: width of the response lights in pixels (default 40)
- *resp_lt_ht*: height of the response lights in pixels (default 40)
- *resp_bt_rad_offset*: offset of the response buttons radius from the response lights radius in pixels (default 60). The radius of the response buttons will be set at the radius of the response lights minus this offset
- *play_bt_rad_offset*: offset of the play buttons radius from the response button radius in pixels (default 60). The radius of the play buttons will be set at the radius of the response buttons minus this offset

CHAPTER

FOUR

STIMULATION FILE

Stimulation files specify the stimuli that will be played on each trial of the test. Figure *Example stimulation file.* shows an example stimulation file.



	A	B	C	D	E	F	G
1	az_angle	elev_angle	sound_file	condition	level	roving	feedback
2	0	0	pink_noises/noise1.wav		65	6	true
3	45	0	pink_noises/noise2.wav		65	6	true
4	90	0	pink_noises/noise3.wav		65	6	true
5	135	0	pink_noises/noise4.wav		65	6	true
6	180	0	pink_noises/noise5.wav		65	6	true
7	225	0	pink_noises/noise6.wav		65	6	true
8	270	0	pink_noises/noise7.wav		65	6	true
9	315	0	pink_noises/noise8.wav		65	6	true
10	0	45	pink_noises/noise9.wav		65	6	true
11	45	45	pink_noises/noise10.wav		65	6	true
12	90	45	pink_noises/noise11.wav		65	6	true
13	135	45	pink_noises/noise12.wav		65	6	true
14	180	45	pink_noises/noise13.wav		65	6	true
15	225	45	pink_noises/noise14.wav		65	6	true
16	270	45	pink_noises/noise15.wav		65	6	true
17	315	45	pink_noises/noise16.wav		65	6	true
18							

Fig. 1: Example stimulation file.

Each row of the file represents a trial. Stimulation files contain the following columns:

- *az_angle*: the azimuth angle at which the sound will be presented (if in mode *speakers*, stimuli will be sent to the corresponding soundcard channel as specified in the parameters file)
- *elev_angle*: the elevation angle at which the sound will be presented (if in mode *speakers*, stimuli will be sent to the corresponding soundcard channel as specified in the parameters file)
- *sound_file*: the path (relative or absolute) to the WAV file to be played
- *condition*: an optional label specifying the experimental condition
- *level*: the *base* sound level (in dB SPL) at which the sound will be presented (this assumes that *sound_source_id* has been correctly calibrated)
- *roving*: a level rove, the actual sound level will be equal to the base level plus a value drawn from a random uniform distribution between +/- the roving level
- *feedback*: if *true*, feedback will be given to the listener at the end of each trial

CHAPTER

FIVE

RESULTS FILES

The main results file is a comma-separated values (CSV) file. Note that the CSV separator is not necessarily a comma, and can be changed in the `General` Preferences tab accessible by clicking `Edit`, and then `Preferences`. The main results file contains a row for each trial with the following fields:

- *listener*: the listener identifier
- *condition*: the experimental condition
- *block*: the block number
- *trial*: the trial number
- *azimuth_angle*: the stimulus azimuth angle, as specified in the parameters file
- *elevation_angle*: the stimulus elevation angle, as specified in the parameters file
- *response_azimuth*: the response azimuth angle
- *response_elevation*: the response elevation angle
- *azimuth_error*: the signed azimuth angle error
- *elevation_error*: the signed elevation angle error
- *azimuth_angle_remapped*: the stimulus azimuth angle remapped to the 0-360° range
- *response_azimuth_remapped*: the response azimuth angle remapped to the 0-360° range
- *azimuth_angle_flip*: the stimulus azimuth angle obtained after *flipping* (reflecting/mirroring) the stimulus positions from the rear to the front
- *response_azimuth_flip*: the response azimuth angle obtained after *flipping* (reflecting/mirroring) the stimulus positions from the rear to the front
- *azimuth_error_flip*: the azimuth error calculated from stimulus and response angles flipped to the front (this is one way to calculate errors while ignoring front-back confusions)
- *front-back*: `[0, 1]` 1 if the listener made a front-back confusion, 0 otherwise
- *sound_file*: the sound file used for the trial
- *base_level* the base level, in dB SPL, for the trial
- *rove*: the level rove range for the trial
- *actual_level*: the actual level at which the stimulus was played
- *date*: the date
- *time*: the time

`sound_source_id` additionally outputs summary files for the all the trials, as well as by experimental condition and/or by block if multiple conditions/blocks are present (and by block x condition). For each of these subdivisions of the results, one summary file gives root-mean-square (RMS) azimuth and elevation errors, and the other front-back error proportions. The RMS error files contain the following fields:

- *listener*: the listener's identifier
- *rms_azimuth_err*: the RMS azimuth error
- *rms_elevation_err*: the RMS elevation error
- *rms_azimuth_err_flip*: the RMS azimuth error calculated after both stimuli and responses positions have been flipped from the rear to the front (this is one way to compute RMS errors while ignoring front-back errors)
- *azr_rms_err_no_FB*: the RMS azimuth error calculated after excluding trials with front-back errors (this is another way to compute RMS errors while ignoring front-back errors). Front-back errors are defined here as cases in which the RMS error is reduced after flipping both stimuli and responses positions to the front

The front-back error files contain the following fields:

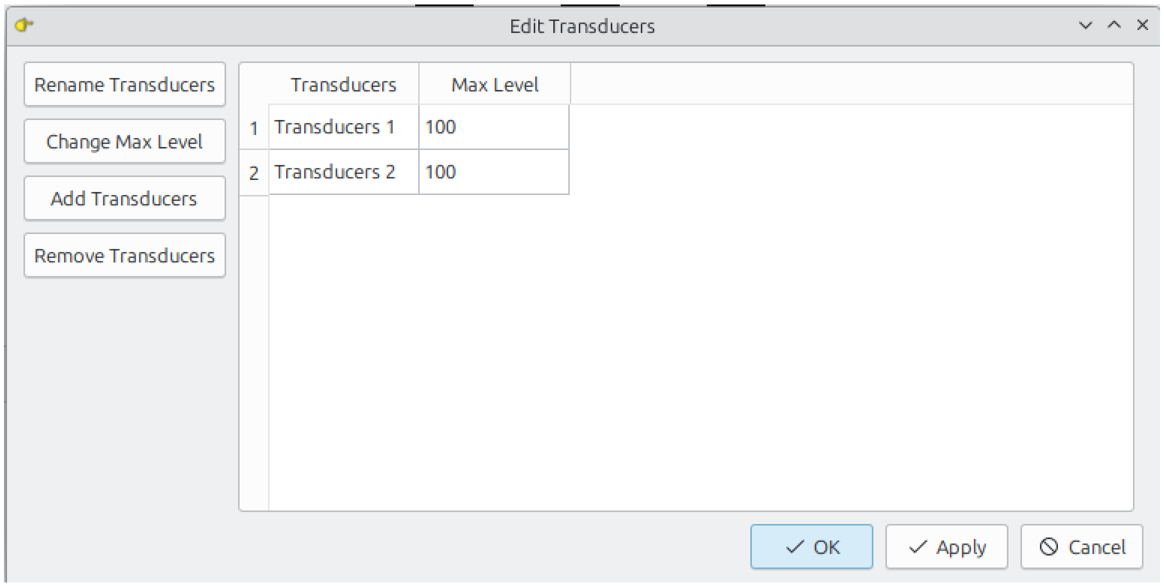
- *listener*: the listener's identifier
- *front_back*: the proportion of front-back errors

CHAPTER

SIX

SOUND LEVEL CALIBRATION

Figure *Edit transducers dialog* shows a screenshot of the `Transducers` dialog which is used for setting calibration values.



	Transducers	Max Level
1	Transducers 1	100
2	Transducers 2	100

Fig. 1: Edit transducers dialog

Most of the fields should be pretty much self explanatory. Using this dialog you can add headphones/speakers models to the transducers database. In *sound_source_id* levels are referenced to the level that would be output by a full amplitude sinusoid (a sinusoid with a peak amplitude of 1). In the `Max Level` field you should enter the level in dB SPL that is output by the transducer for a full amplitude sinusoid . However, getting reliable readings for a pure tone with an SPL meter is difficult, therefore, typically a noise is used for calibrating loudspeakers.

The procedure I normally use for calibrating loudspeakers is to save on disk a noise stimulus as a wav file. I filter the noise within the operating range of the SPL meter (usually around 0.05 to 8 kHz). The noise level needs to be reasonably high as to avoid signal-to-noise ratio issues, but not so high as to cause distortions or damage your hearing in the measurement process. Once I've found a reasonable level, by trial and error, I measure the actual level with an SPL meter held at the position where the listener head would be located relative to the loudspeaker during the experiment, and note it down.

We can measure the root-mean-square (RMS) level of the WAV file with the noise used for calibration, let's call it RMS_{noise}. A full amplitude sinusoid has an RMS amplitude of $1/\sqrt{2} = 0.707$. The difference in dB between the level of a sinusoid at max amplitude and our calibration noise will be equal to:

$$\Delta_{dB} = 20\log_{10}\left(\frac{1/\sqrt{2}}{RMS_{noise}}\right)$$

Therefore, if our calibration noise had a level (measured with the SPL meter) of x dB SPL, a sinusoid at max amplitude would have a level of:

$$maxlev = x + \Delta_{dB}$$

this is the value that you need to enter in the `Max Level` field of the transducers calibration table for the loudspeakers in question.

CHAPTER

SEVEN

INDICES AND TABLES

- genindex
- modindex
- search