Perceptually-Constrained Spotforming-Inspired Spatial Audio - README

Dimme de Groot

March 5, 2024

1 Introduction

In this README, we explain the data and code concerning the paper *Perceptually-Constrained Spotforming-Inspired Spatial Audio*[1].

- We consider using the supplied MATLAB code, and in particular focus on recreating Figures 2, 3 and 4 of the paper. This corresponds to the files given in `FilesA - Reproduce Simulation Results.zip`.
- We explain the table containing the listening-test results and give some useful code for having a look at the results. This code can also be used to reproduce Tables II and III. This corresponds to the files given in `FilesB - Data Subjective Experiments.zip`.

[1] A preprint of this paper is supplied together with this README. The paper is currently under submission for IEEE Transactions on Audio, Speech and Language Processing.

2 Using the code to reproduce Figures 2, 3 and 4

Before using the code, please consider that the code is not well tested for different settings from those used in the paper. I.e. using different sample frequencies, window lengths, etc. might be problematic. You should be able to modify the room, audio signals, and distribution parameters used without difficulties. We do not attempt to be complete in our description of the code.

2.1 File structure

The code is structured in a number of folders:

- `Audio`: this folder contains audiofiles (sampled at 8 kHz!) for which you want to compute the playback signals. The white Gaussian noise signal used as s_{ref} in the paper is provided.
- `functions`: this folder contains some functions which are needed to perform the computations.
  - `fnc_get_sx`: this function takes as input a room object (see below) and creates a list of image sources and the corresponding location and reflection coefficients. This is returned as a list indexed by the loudspeaker number.
  - `fnc_nearest_neighbour`: this function takes as input a room object (see below) and returns the nearest neighbouring loudspeaker w.r.t. the virtual source. This nearest neighbour is the neighbour nearest in angle w.r.t. the expected center of the listeners head location $\mathbf{x}_h$.
  - `P_par`: this function takes as input a par_set object and the input signal frame and gives as output the masking curve (in both SPL dB and in 'normal' units) and the inverse masking curve.
- `Objects`: this folder contains the objects used. These are the following:
  - `Par_Set`: this object is used for the Par-measure. It implements the filters needed for the Par measure and performs the calibration procedure.
  - `Room`: this object defines the room: the receiver location $\mathbf{x}_r$, the physical loudspeaker locations $\mathbf{x}_i$, the virtual source location, the room size and the reflection coefficients. A crude estimate of the T_{60} time is provided as well. In the `Room` object, the virtual source location is described as a *location*, and not as a *direction*. The direction is computed from this location in all code where the virtual source direction is needed. **The default values are those used in the simulation section of the paper.**
  - `Settings`: a simple object keeping track of the settings, such as the integration bounds and the frame-length. Here, also the parameterisation $\kappa_{\mathcal{V}}, \mu_r, \sigma_r$ of the probability distributions is specified. **The default values are those used in the paper.**
- `Results`: the results when running the code are stored in this folder. The results needed to recreate Fig. 4 are already provided and stored in this folder.
- `step0_calibrate`: this folder contains the code which is relevant when choosing a room and distribution. Namely
  - `fnc_plot_room`: used when plotting the room.
  - `step0a_visualise_room`: used to plot the room.
  - `step0b_visualise_weighting`: used to visualise the probability distribution.
- `step1_PSD_matrices`: this folder contains the code required to numerically compute the spatial correlation matrices $\mathbf{R}_B$ and $\mathbf{R}_D$. In the code `A` corresponds to $\mathbf{R}_B$ and `B` corresponds to $\mathbf{R}_D$. Sorry for this confusing notation.
  - `step1a_compute_PSD_integral`: this code computes the integrals required for $\mathbf{R}_{\mathcal{B}}$ and $\mathbf{R}_{\mathcal{D}}$. The results are stored (per loudspeaker) in the subdirectory `step1a_compute_PSD_integral/Data/<REGION_LOUDSPEAKER>.mat`, where `<REGION_LOUDSPEAKER>.mat` is the region (`A` or `B` for $\mathcal{B}$ and $\mathcal{D}$, respectively and `LOUDSPEAKER` is the loudspeaker number (1 to $N_s + 1$, since the virtual source is also considered).

- `step1b_combine_PSD_integral`: this code combines the results computed by `step1a_compute_PSD_integral` and stores the results in `step1a_compute_PSD_integral/Data/data_raw.mat`.
- `step1c_compute_PSD`: this code computes $\mathbf{R}_B$ and $\mathbf{R}_D$ from $\mathbf{R}_\mathcal{B}$, $\mathbf{R}_\mathcal{D}$ and σ^2_{num}. The latter is found in the `Settings` object. The results are stored in `step1a_compute_PSD_integral/Data/data_proc.mat`.
- `step1d_decompose_PSD`: this function performs the decomposition needed for the convex relaxation, i.e. it computes $\mathbf{R}$ and $\mathbf{L}$. It also normalises $\mathbf{L}$ by its Frobenius norm. That is, it implements α_1 of the paper. The results are stored in `step1a_compute_PSD_integral/Data/data_res.mat`.
- The function `fnc_main`: this function loops through the different frequency bins and computes the mean RIR (unused) by integrating over `fnc_integrandA` and `fnc_integrandB` and the covariance matrices by integrating over `fnc_integrandAstd` and `fnc_integrandBstd`. The latter two are implementations of the integrand of Eq. (22) of the paper. The former two are not used in the paper, but can be considered as the integrand of Eq. (5) of the paper, but with the transfer $\hat{h}$ instead of the audio signal $\hat{s}$.

- `step2_sound`: this folder contains the code used to implement NN, GEV and the novel algorithm.
  - `main_1_nn`: this file implements NN (nearest neighbour) and stores the results in the `Results` folder.
  - `main_2_gev`: this file implements the GEV-based algorithm and stores the results in the `Results` folder.
  - `main_3a_transfer`: this file creates some stuff needed to perform the novel algorithm. Among others, the zero-padded DFT matrix, the analysis and synthesis windows, the `par_set` object and the distortion-constrained loudspeaker ε is defined and stored in the `Results` folder.
  - `main_3b_novel`: this file implements the novel algorithm. The user needs to define d_{par} (`dPar`) and α_2 (`lambda`). The results are stored in the `Results` folder.

- `step3_simulation` This folder is used to compute things for evaluating the results. In particular, it uses Habets implementation of the mirror image source method (MISM) to compute the RIRs to a number of points in the highlighted region and subsequently construct the plots. The folder contains the following items
  - `fnc_habets`: this functions takes as input a `Room` object and a `Settings` object and returns a list of RIRs from each of the loudspeakers (stored in `room.S`) to the receiver location (stored in `room.R`). It uses `rir_generator.mexa64` to do so, see below.
  - `fnc_nearest_neighbour`: this is a copy of the function with the same name in the `functions` folder described above.
  - `fnc_subplot_image_1`: this function is handy for plotting and used by `step3b_octavefilters`.
  - `rir_generator.mexa64`: this is the RIR generator from Habets, it is not provided by us. You can obtain it from `https://github.com/ehabets/RIR-Generator`. Alternatively, we provide the RIRs needed to recreate Figure 4 in the `step3_simulation/Data` folder.
  - `step3a_compute_transfer_functions`: this file is used to compute the RIR to a number of points using `rir_generator.mexa64`. The resulting RIRs are stored in the `Data` folder.
  - `fnc_subplot_image_1`: this function is handy for plotting and used by `step3b_octavefilters`.
  - `step3b_octavefilters`: this file uses the RIRs stored in the `Data` folder and computes the energy per octaveband per RIR for a given set of playback signals (as stored in the `../Results` folder). This file reproduces Fig. 4 of the paper.

2.2 Using the code: reproducing Figure 2 and 3

You can run `step0_calibrate/step0a_visualise_room.m` to reproduce Figure 3. In case you want to modify the room, simply modify the `Objects/Room.m` object.

You can simply run `step0_calibrate/step0b_visualise_weighting.m` to reproduce Figure 2. In case you want to modify the distribution parameters, simply modify the `Objects/Settings.m` object (`mu_r`, `sigma_r`, `kappaA`, `kappaB`), the parameters `muA` and `muB` are computed from `room.R` and the virtual source direction (last row of `room.S`).

2.3 Using the code: computing playback signals

As mentioned in the paper, we use CVX equipped with MOSEK to solve the optimisation problems, see `https://cvxr.com/cvx/download/`. The NN and GEV algorithm do not need to perform convex optimisation. However, to use `step2_sound/main_3b_novel`, you will need to install CVX. It might also work without MOSEK installed, but the accuracy and runtimes will likely differ.

Note that we provide all playback signals required to reproduce Fig. 4. So, if that is your goal, you do not need to recompute them.

To compute the playback signal for the nearest neighbour algorithm, simply provide the path to your audiofile (example is given for `"../Audio/white_noise.wav"`) and run `step2_sound/main_1_nn.mat`. The result will be stored in the `Results` folder.

To compute the playback signals for the GEV-based algorithms and the novel algorithms, you first need to compute the spatial covariance matrices. The covariance matrices are computed by running `step1_PSD_matrices/step1a_compute_PSD_integral` up to `step1d_decompose_PSD`. Note that we use a `parfor` loop, so Matlabs parallel computing toolbox is required (or you can modify the code). If you want to compute the covariance matrices for a different room or for different distribution parameterisations, simply modify the corresponding files in the Objects folder.

Now you can simply run `main_2_sound/main_2_gev.m` to compute the GEV-based algorithms playback signals. The audio path should be specified (example is given for `"../Audio/white_noise.wav"`).

To compute the novel algorithms playback signals, first run `main_2_sound/main_3a_transfer.m` and then run `main_2_sound/main_3b_novel`. The audio path should be specified in the latter (example is given for `"../Audio/white_noise.wav"`).

2.4 Using the code: reproducing Figure 4

To reproduce Figure 4, first run `step3_simulation/step3a_compute_transfer_functions.m` and then run `step3_simulation/step3b_octavefilters.m`. The former computes the required RIRs and the letter plots Fig. 4. Note that, if you didnt modify any of the parameters, the `step3_simulation/Data` folder already contains the RIRs, so it is not needed to run `step3_simulation/step3a_compute_transfer_functions.m`. Additionally, note that `step3_simulation/step3b_octavefilters.m` requires the paths to the computed audiofiles. Currently, the paths needed to reproduce Fig. 4 are given. If you want to use the same colormap, please visit `https://www.fabiocrameri.ch/colourmaps/`. Here, the perceptually uniform and colour-vision-deficiency friendly colormaps can be downloaded, along with some explanation.

3 Subjective experiment results

We provide two `.mat` files concerning the results of the subjective experiments. Additionally, some functions are provided plotting the results. Below, we first briefly describe the files. We then explain the structure of the tables.

3.1 Provided files

- `Results/table_loc.mat`: this is the table containing the results of the localisation experiment.
- `Results/table_statistics.mat`: this is the table containing some statistics, such as confidence and intelligibility ratings. We removed gender information from this table.
- `fnc_cart2sph.m`: this function converts cartesian coordinates to spherical coordinates.
- `step1_statistics.m`: this function prints the statistics. It effectively reproduces Table II.
- `step2a_azimuth_elevationMeas.m`: this function reports the errors for a given virtual source direction. Thus, it effectively reproduces Table III. Additionally, it plots the mean directions reported, and it plots the individual responses.
- `step2b_azimuth_elevationTrain.m`: this function plots the responses of the users to the training signals. Note that all training signals are single loudspeakers playing back a signal.

3.2 The .mat tables

Below, the content of the tables is described. When observing the tables, you will sometimes notice a `NaN` value. This is because sometimes the listeners did not answer all questions and because sometimes the post-its (used to measure the location the listener pointed at) fell of the crepe-paper.

The table: `Results/table_statistics.mat`

This is a 780×11 matrix. The columns (ordered) indicate the following:

1. Medically diagnosed hearing impairment (1 yes; 2 no; 3 would rather not say)
2. Age range (1 younger than 19; 2 19-30; 3 30-40; 4 41-60; 5 61 or older; 6 would rather not say)
3. Native English speaker (1 yes; 2 no; 3 would rather not say)
4. Self-reported English level (1 beginner; 2 pre-intermediate; 3 intermediate; 4 near fluent; 5 fluent; 6 would rather not say)
5. Reported confidence (1 to 5, where higher is more confident)
6. Reported quality (1 to 5, where higher is higher quality)
7. Reported intelligibility (1 to 5, where higher is more intelligible)
8. Virtual source direction (1 to 11)
9. Algorithm (1 NN; 2 GEV; 3 Novel $\alpha_2 = 0.1$; 4 Novel $\alpha_2 = 1$)
10. Gender of speaker, i.e. was the segment from the TIMIT database a female or a male speaker (1 female; 2 male).
11. Number of repetitions, how often did the listener request the signal to be repeated?

The table: `Results/table_loc.mat`

This is a 936×15 matrix. The columns (ordered) indicate the following:

1. Listener number, replaced by all -1's (i.e. not informative).
2. Measured x-location, centered at `room.R` ($\mathbf{x}_h$).
3. Measured y-location, centered at `room.R` ($\mathbf{x}_h$).
4. Measured z-location, centered at `room.R` ($\mathbf{x}_h$).
5. Virtual source direction (1 to 11)
6. Algorithm (1 NN; 2 GEV; 3 Novel $\alpha_2 = 0.1$; 4 Novel $\alpha_2 = 1$)
7. Gender of speaker, i.e. was the segment from the TIMIT database a female or a male speaker (1 female; 2 male; 3 part of the training segments). All training segment speakers were female as well. For the training segments, the algorithm and virtual source direction column are ignored.
8. Target x-location, centered at `room.R` ($\mathbf{x}_h$). I.e. a x-location corresponding to a virtual source direction.
9. Target y-location, centered at `room.R` ($\mathbf{x}_h$). I.e. a y-location corresponding to a virtual source direction.
10. Target z-location, centered at `room.R` ($\mathbf{x}_h$). I.e. a z-location corresponding to a virtual source direction.
11. Nearest Neighbour (reference loudspeaker ε) x-location, centered at `room.R` ($\mathbf{x}_h$)
12. Nearest Neighbour (reference loudspeaker ε) y-location, centered at `room.R` ($\mathbf{x}_h$)
13. Nearest Neighbour (reference loudspeaker ε) z-location, centered at `room.R` ($\mathbf{x}_h$)
14. Medically diagnosed hearing impairment (1 yes; 2 no; 3 would rather not say)
15. Age range (1 younger than 19; 2 19-30; 3 30-40; 4 41-60; 5 61 or older; 6 would rather not say)