





OBF-MP toolbox

Orthonormal Basis Function - Matching Pursuit algorithm

available for download at ftp://ftp.esat.kuleuven.be/stadius/gvairett/OBF-MP.zip

Author: GIACOMO VAIRETTI email: giacomo.vairetti@esat.kuleuven.be
Promotors: MARC MOONEN, TOON VAN WATERSCHOOT date: JUNE 23, 2016

1 Introduction 1
2 OBF-MP algorithm 2
2.1 Definition of variables 2
2.2 Initialization 3
2.3 Iteration 3
2.4 Asymptotic Computational Complexity 4
3 Content 5
3.1 `OBF_MP_alg.m` (function) 5
3.2 `poleGrid.m` (function) 5
3.3 `computeMixKautz.m` (function) 5
3.4 `OBF_BU_alg.m` (function) 5
3.5 `firstPeak.m` (function) 5
3.6 `simulations.m` (script) 6
3.7 `showSimulResults.m` (script) 6
3.8 `showPrecomputedResults.m` (script) 6
3.9 `simulations_trial.m` (script) 6
3.10 Third-party code 7
3.11 Data 7
4 Disclaimer 9
5 Acknowledgements 9

1 INTRODUCTION

In this toolbox, the OBF-MP algorithm proposed in [1] and further analyzed in [2] is implemented in MATLAB®, and some scripts are provided for reproducing the simulation results shown in [2]. In the following, only an overview of the algorithm is provided. For more information and details, please refer to [2]. OBF-MP is an optimization algorithm for solving the problem of compactly representing a signal by a linear combination of a subset of elements (called predictors) from overcomplete bases, by using OBF models. The particular OBF model used is the so-called mixed-Kautz model, shown in Fig. 1 for n real poles and m pairs of complex conjugate poles, which is a combination of the Takenaka-Malmquist model for real poles and the Kautz model for complex poles. The algorithm is based on the well-known iterative greedy algorithm known as Orthogonal Matching Pursuit (OMP), with the difference that the matrix of candidate predictors has to be up-

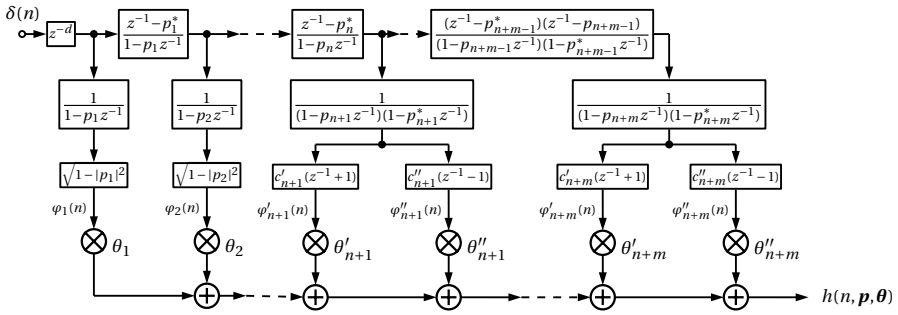

Figure 1: The mixed-Kautz model for n real poles and m pairs of complex-conjugate poles, with normalization parameters $c'_i = |1-p_i|\sqrt{1-|p_i|^2/2}$ and $c''_i = |1+p_i|\sqrt{1-|p_i|^2/2}$.

dated at each iteration, accordingly to the OBF model structure. At each iteration, the algorithm selects the predictor(s) among those in the matrix of candidates that best approximate the target room impulse response (RIR) (i.e. the one(s) with the largest correlation with the RIR), which is orthogonal to the predictors already selected to be included in the basis.

2 OBF-MP ALGORITHM

The problem of sparse linear approximation of a signal consists in finding a compact representation by a combination of functions taken from an overcomplete basis. These functions are usually called predictors or atoms, which altogether form a basis, sometimes called dictionary.

Our approach aims at finding a sparse representation of a target RIR as a linear combination of OBFs. This section describes an OMP-based greedy algorithm, which is termed here OBF-MP [1]. At each iteration of the OMP algorithm, the predictor that has the highest correlation with the current residual response is selected. The problem in the OBF-MP algorithm, given that OBFs are defined by previous poles in the structure, is how to define a dictionary of candidate predictors. As a consequence, the dictionary has to be updated at each iteration using the predictor selected at the previous iteration. The advantage of OBF-MP over the conventional OMP algorithm is that the projection onto the orthogonal space is not necessary. The predictors, in fact, are already orthogonal to each other by construction. This also ensures that the algorithm does not contain any matrix inversion, thus avoiding ill-conditioning problems. Moreover, since the candidate predictors are orthogonal to the predictors selected at previous iterations, computing the correlation with the current residual response is equivalent to computing the correlation with the target response. Another consequence of orthogonality is the scalability of the algorithm, for which the linear coefficients do not have to be recomputed when a new iteration is performed, or when a model with a smaller number of parameters is obtained by truncation. Also, there is no need to predetermine the number of coefficients.

2.1 Definition of variables

The purpose of the OBF-MP algorithm is to iteratively compute a sparse approximation $\hat{\boldsymbol{h}}$ of a target RIR $\boldsymbol{h}$ of length N samples as a linear combination of length-N impulse responses of OBFs. The OBFs are selected from a dictionary $\boldsymbol{\Phi}_k$ of candi-

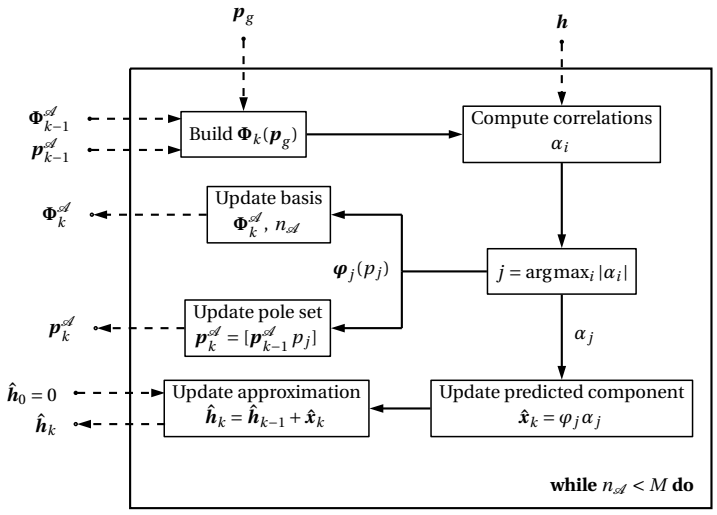

Figure 2: The OBF-MP algorithm block diagram. Inbound dashed lines represent initial conditions and inputs, while outbound dashed lines represent outputs.

date predictors $\boldsymbol{\varphi}_i$ $(i = 1,\ldots,D)$ and included in the basis $\boldsymbol{\Phi}^{\mathscr{A}}_k$. At each iteration k, D candidate predictors $\boldsymbol{\varphi}_i$, orthogonal to the predictors in the current basis $\boldsymbol{\Phi}^{\mathscr{A}}_{k-1}$ constructed with the current set of active poles $\boldsymbol{p}^{\mathscr{A}}_k$, are built from G poles placed arbitrarily in a grid $\boldsymbol{p}_g$ inside the unit disc. The matrix $\boldsymbol{\Phi}_k$ has dimensions $N \times D$, with $D = n+2m$ where n and m denotes respectively the number of real poles and complex-conjugate poles in the grid $\boldsymbol{p}_g$, so that $G = n+m$.

2.2 Initialization

The OBF-MP algorithm is described in detail below, and a graphical representation is depicted in Fig. 2. First, a grid of G candidate poles $\boldsymbol{p}_g$ is defined, similarly to [3], with poles distributed accordingly to a desired frequency resolution or prior knowledge about the system. At the first iteration, the current basis $\boldsymbol{\Phi}^{\mathscr{A}}_0$ and the set of active poles $\boldsymbol{p}^{\mathscr{A}}_0$ are empty (with the number of predictors in the basis $n_{\mathscr{A}} = 0$). Also the target approximation vector $\hat{\boldsymbol{h}}_0$, is initially set to zero.

2.3 Iteration

At each iteration k, the matrix of candidate predictors $\boldsymbol{\Phi}_k(\boldsymbol{p}_g)$ is updated according to the mixed-Kautz structure in Fig. 1, where two real-valued responses are generated for each complex pole pair. The predictor(s) in $\boldsymbol{\Phi}_k$ with maximum absolute correlation α_i with the target RIR vector $\boldsymbol{h}$ is selected and added to the basis $\boldsymbol{\Phi}^{\mathscr{A}}_k$, while the corresponding pole is included in the set of active poles $\boldsymbol{p}^{\mathscr{A}}_k$. The correlation for real and complex poles is computed in two different ways. For real poles, the correlation is taken as the projection of $\boldsymbol{h}$ onto the predictor $\boldsymbol{\varphi}_i$ $(\alpha_i = \boldsymbol{\varphi}_i^T \boldsymbol{h})$. For a pair of complex-conjugate poles $\{p_i, p_i^*\}$ the correlation is chosen as the projection of $\boldsymbol{h}$ on the plane defined by predictors $\boldsymbol{\varphi}'_i$ and $\boldsymbol{\varphi}''_i$ (see Fig. 3), and is given by

$$\alpha_i = \sqrt{\alpha_i'^2 + \alpha_i''^2} = \sqrt{(\boldsymbol{\varphi}_i'^T \boldsymbol{h})^2 + (\boldsymbol{\varphi}_i''^T \boldsymbol{h})^2}. \tag{1}$$

The kth predicted component $\hat{\boldsymbol{x}}_k$ is obtained from the last added predictor(s) using the maximum correlation α_s as regression coefficient (with $s = \text{argmax}_i\,|\alpha_i|$),

Algorithm 1 OBF-MP algorithm

1: $\boldsymbol{p}_g = \{p_1, \ldots, p_G\}$ % Define poles in the pole grid
2: $\boldsymbol{\Phi}_0^{\mathscr{A}} = \emptyset, n_{\mathscr{A}} = 0$ % Initialize set of active predictors (basis)
3: $\boldsymbol{p}_0^{\mathscr{A}} = \emptyset$ % Initialize the set of active poles
4: $\hat{\boldsymbol{h}}_0 = \boldsymbol{0}$ % Initialize target approximation vector
5: $k = 1$ % k: iteration counter
6: **while** $n_{\mathscr{A}} < M$ **do** % M: desired number of OBFs
7: Build $\boldsymbol{\Phi}_k(\boldsymbol{p}_g)$ % Build matrix of candidate predictors $\boldsymbol{\varphi}_i$
8: $s = \arg\max_i |\alpha_i|$ % Find $\boldsymbol{\varphi}_i$ max. correlated with $\boldsymbol{h}$
9: $\boldsymbol{p}_k^{\mathscr{A}} = [\boldsymbol{p}_{k-1}^{\mathscr{A}}\ p_s]$ % Add selected pole to active pole set
10: **if** p_s real **then** % Update basis and predicted component
11: $\boldsymbol{\Phi}_k^{\mathscr{A}} = [\boldsymbol{\Phi}_{k-1}^{\mathscr{A}}\ \boldsymbol{\varphi}_s]$, $n_{\mathscr{A}} = n_{\mathscr{A}} + 1$
12: $\hat{\boldsymbol{x}}_k = \boldsymbol{\varphi}_s \alpha_s$
13: **else if** p_s complex **then**
14: $\boldsymbol{\Phi}_k^{\mathscr{A}} = [\boldsymbol{\Phi}_{k-1}^{\mathscr{A}}\ \boldsymbol{\varphi}_s'\ \boldsymbol{\varphi}_s'']$, $n_{\mathscr{A}} = n_{\mathscr{A}} + 2$
15: $\hat{\boldsymbol{x}}_k = [\boldsymbol{\varphi}_s' \boldsymbol{\varphi}_s''][\alpha_s' \alpha_s'']^T$
16: **end if**
17: $\hat{\boldsymbol{h}}_k = \hat{\boldsymbol{h}}_{k-1} + \hat{\boldsymbol{x}}_k$ % Update target approximation vector
18: $k = k + 1$
19: **end while**

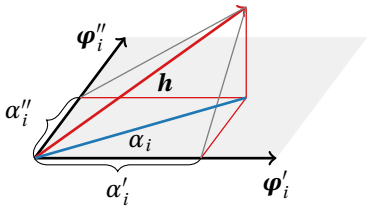

Figure 3: Graphical interpretation of the correlation between the target RIR vector $\boldsymbol{h}$ and the predictors of a pair of complex-conjugate poles $\{p_i, p_i^*\}$.

by $\hat{\boldsymbol{x}}_k = \boldsymbol{\varphi}_s \alpha_s$, if the selected pole is real, or by $\hat{\boldsymbol{x}}_k = [\boldsymbol{\varphi}_s' \boldsymbol{\varphi}_s''][\alpha_s' \alpha_s'']^T$, if complex. The current target approximation vector $\hat{\boldsymbol{h}}_k$ is obtained by adding the predicted component $\hat{\boldsymbol{x}}_k$ to the previous target approximation vector $\hat{\boldsymbol{h}}_{k-1}$. The algorithm terminates when the desired number M of predictors in the basis is reached. Alternatively one can also terminate the algorithm when the approximation error falls below a desired value.

2.4 Asymptotic Computational Complexity

Here the asymptotic computational complexity of the OBF-MP algorithm is analyzed, assuming for simplicity that only complex poles are included in the pole grid. With reference to Algorithm 1, there are two operations that determine the asymptotic behavior of the algorithm. The building of the matrix $\boldsymbol{\Phi}_k$ of candidate predictors (step 7) at each iteration is the most demanding operation, which involves the generation of D predictors of length N, which sums up to a complexity in the order of $\mathcal{O}(3ND)$ multiplications. The second operation to consider is the computation of the correlation coefficients (step 8), which is a multiplication of the matrix $\boldsymbol{\Phi}_k$ with a vector of length N ($\mathcal{O}(ND)$ multiplications). Vector updates and other operations add little to the computational burden. After $k = M/2$ iterations, the overall complexity of the OBF-MP algorithm is then $\mathcal{O}(2MND)$ multiplications, i.e. linearly proportional to the three variables considered.

3 CONTENT

This section lists the content of the OBF-MP toolbox. Each function, script or data file is defined in the following, while the details can be found in the preamble of each file, and in [2].

3.1 `OBF_MP_alg.m` (function)

`OBF_MP_alg.m`

Scalable automatic model-building algorithm for the sparse approximation of a target signal by means of orthonormal basis function (OBF) models. The OBF-MP algorithm provides an optimal (in the LS sense) approximation of the target signal given by the OBF model of order M (i.e the number of functions in the basis) for a given input vector. See the function preamble for further information.

3.2 `poleGrid.m` (function)

`poleGrid.m`

Define a grid of pole distributed on the unit disc, with different distributions of the angle and of the radius of the poles. This function `grd = poleGrid(M1,M2,lrad,urad,lang,uang,dist,fs)` defines a grid of poles distributed over the unit disc accordingly to the parameters defined in the input, i.e. the number of possible radii `M1` in the grid in the range `lrad-urad`, the number of possible angles `M2` in the frequency range `lang-uang` in radians, the type of distribution `dist` of the radius and angle (uniform, logarithmic, Bark), and the sampling frequency `fs` (needed for Bark scale distributions).

3.3 `computeMixKautz.m` (function)

`computeMix Kautz.m`

Model building procedure for the OBFs of a mixed-Kautz model structure as the one in Fig. 1, given a set of poles and an input signal. This function implements the mixed-Kautz OBF model structure. Given a set of poles, the function build a matrix which columns are the impulse response of the basis functions built upon the poles to the input. See [1], [2] or [4] for a description of the model structure.

3.4 `OBF_BU_alg.m` (function)

`OBF_BU_alg.m`

This function implements the BU method used in [4] for the approximation of a target signal vector for a given input signal vector as a linear combination of OBFs. The target signal vector is warped according to the Bark scale (see Section 3.10). The function also returns the poles selected by the BU method (Ref.[5]), the OBFs vectors and the linear regression coefficients, as well as the number of selected solution with at least one pole outside the unit circle(in which case the first stable solution with lower conversion error is selected).

3.5 `firstPeak.m` (function)

`firstPeak.m`

This function estimates the delay of a set of RIRs stored in a matrix, in order to determine the starting point of the response. The first peak of the RIR,considered as its starting point, is detected based on a threshold (percentage [0,1] w.r.t. the maximum absolute amplitude of the RIR).

3.6 `simulations.m` (script)

`simulations.m (not advised)`

The script runs the simulations which results are shown in [2]. A set of RIRs used as target responses were taken from three publicly available databases, namely MARDY [6], SMARD [7], and MIRD [8]. The OBF-MP algorithm is used to obtain a sparse approximation of the target RIRs with different values of the model complexity. For comparison, an approximation with the same model order is obtained with the warped-BU method in [4] and with the all-zero and pole-zero models. The simulation results can be plotted as for Fig. 8 and Fig. 9 in [2] by running the script `showPrecomputedResults.m` (3.7), where the precomputed results are loaded. It is not advisable to run the complete simulation, as it might be very time consuming (*over 2 hours for each database of 8 RIRs*, on a quad-core laptop). A trial version of the simulation that can be run in a reasonable period of time is provided in `simulations_trial.m` (3.9).

3.7 `showSimulResults.m` (script)

`showSimul Results.m`

The script generates the simulation results shown in [2] (Fig. 8-9). The simulation can be performed by running the script `simulations.m` (3.6). Alternatively, the simulation output is already precomputed in `Results` (3.11) and can be loaded using 3.8. The results of the approximation error (NMSE) for each RIR in the different databases are averaged and the approximation error for the different models vs. the model complexity (i.e. the number of model parameters per RIR) are shown in different time and frequency intervals (see Section VI in [2]). The results shown in Fig. 9 for the filter complexity are obtain by scaling the results in Fig. 8 for the model complexity for the OBF and PF models according to Table I in [2]. It is possible to visualize the results for different databases. See Table II in [2] for further details.

3.8 `showPrecomputedResults.m` (script)

`showPrecomputed Results.m`

The script generates the simulation results shown in [2] (Fig. 8-9). The simulation output is already stored in `Results` (3.11). The results of the approximation error (NMSE) for each RIR in the different databases are averaged and the approximation error for the different models vs. the model complexity (i.e. the number of model parameters per RIR) are shown in different time and frequency intervals (see Section VI in [2]). The results shown in Fig. 9 for the filter complexity are obtain by scaling the results in Fig. 8 for the model complexity for the OBF and PF models according to Table I in [2]. It is possible to visualize the results for different databases. See Table II in [2] for further details.

3.9 `simulations_trial.m` (script)

`simulations_ trial.m`

The script runs the simulations which results are shown in [2], but on a single database and with a smaller number of model parameters (*running time 10'*). A set of 8 RIRs used as target responses are taken from the MARDY [6] database. The OBF-MP algorithm is used to obtain a sparse approximation of the target RIRs with different values of the model complexity. Also the plots for the approximation error as in Fig. 8 and Fig. 9 in [2] are generated at the end of the trial simulation.

3.10 Third-party code

The following functions have been used for estimating the poles for the mixed-Kautz model by means of the warped BU method proposed in [4]. The functions are provided in the toolbox or are available for downloading, together with additional information, at the websites of the Signal Processing and Acoustics Department of Aalto University, Finland (http://www.acoustics.hut.fi/).

braun1.m (function)

braun1.m

BU method - Kautz filter pole position optimization. This function is a part of the Kautz filter toolbox by Tuomas Paatero (http://www.acoustics.hut.fi/software/kautz/kautz.htm). An iterative method to optimize the allpass (AP) structure defining a real Kautz filter utilizing a FIR-to-IIR conversion method proposed by Brandenstein and Unbehauen, 1998 (Ref.[5]).

barkwarp.m (function)

barkwarp.m

The function computes an optimal warping coefficient vs sample rate in Hertz to approximate the psychoacoustic Bark scale approximation by J.O. Smith and J.S. Abel (1999) [9]. This function is a part of WarpTB - a Matlab toolbox for warped signal processing (http://www.acoustics.hut.fi/software/warp/), by Matti Karjalainen, Aki Härmä (Ref.[10]).

bark_impres.m (function)

bark_impres.m

The function computes a warped version of a signal or impulse response. This function is a part of WarpTB - a Matlab toolbox for warped signal processing (http://www.acoustics.hut.fi/software/warp/), by Matti Karjalainen, Aki Härmä (Ref.[10]).

lin2bark.m (function)

lin2bark.m

The function computes the frequencies warped according to the Bark scale and the warping parameter (as barkwarp.m). It is used in `poleGrid.m` when the poles are distributed in the angle according to the Bark scale. `[WB, G] = lin2bark(W, Fs)` returns the parameter `G` such that the conformal map $z \rightarrow$ `(z + G)/(z*G + 1)` closely approximates a Bark frequency warping for the input sample rate `Fs`. The input frequencies `W` are warped according to the conformal map generated by Fs to form Bark frequencies WB. This function is a part of the code by J.O. Smith and J.S. Abel (https://ccrma.stanford.edu/~jos/bbt/bbt.html, Ref.[9]).

3.11 Data

The RIR databases can be downloaded directly using the following functions. An example of how to download and load the databases is provided in 3.6.

MARDY (database)

MARDY

`downloadMARDY.m` This function downloads the MARDY database [6] (http://www.commsp.ee.ic.ac.uk/~sap/uploads/data/MARDY.rar). As an input, the path to a software for decompressin rar archives (tested with WinRar and 7-zip) should be provided. Please use the format provided in this example: `"C:\Program Files\7-Zip\7z.exe"`.

`loadMARDY.m` The function returns a cell containing the RIRs from the MARDY database. The RIRs are selected based on the information provided about the positions of the microphone arrays in the room and the channel chosen (from 1 to 8). Also the desired length (in seconds) of the RIRs should be provided. See [6] or visit http://www.commsp.ee.ic.ac.uk/~sap/resources/mardy-multichannel-acoustic-reverberation-database-at-york-database/ for more information. The function returns a struct containing the DB cell and other useful information, such as the volume and surface of the room, the mixing time, the reverberation time and other support variables.

SMARD (database)

SMARD

`downloadSMARDsingle.m` This function downloads the SMARD database [7] (http://www.smard.es.aau.dk/) containing the measurements taken with a single microphone. As an input, the type of source, its position and the position of the microphone should be provided. Visit the website to see the possible configurations.

`loadSMARDsingle.m` The function returns a cell containing the RIRs from the SMARD database containing the measurements taken with a single microphone. As an input, the type of source, its position and the position of the microphone should be provided, together with the desired length (in seconds) of the RIRs. The function returns a struct containing the DB cell and other useful information, such as the volume and surface of the room, the mixing time, the reverberation time and other support variables.

`loadSMARDataRIR.m` The function was adapted from the one in http://shares01.portal.aau.dk/smard/code/lib/loadSmarData.zip to load a RIR and associated setup from the SMARD database.

MIRD (database)

MIRD

`downloadMIRDxxx.m` This function downloads the MIRD databases (xxx denotes the reverberation time (160, 360, 610 ms) and return the path in which the RIRs are extracted. As an input, the microphone spacing should be provided. See [8] or visit http://www.ind.rwth-aachen.de/de/forschung/tools-downloads/multichannel-impulse-response-database/ for the possible configurations.

`loadMIRD.m` The function returns a cell containing the RIRs from the MIRD database. As an input, the reverberation time, the positions of the loudspeaker and the channel of the microphone array should be provided, together with the desired length (in seconds) of the RIRs. The function returns a struct containing the DB cell and other useful information, such as the volume and surface of the room, the mixing time, the reverberation time and other support variables.

`downloadMIRDsupportPack.m` The function downloads a package used to extract the database and other useful functions for processing data. The path of the directory where the package is extracted is given as an output. The SP package seems to have some problems with newer versions of MATLAB (e.g. 2016a).

Results (folder)

Results

The folder contains the precomputed results of the simulations resulting from the script `simulations.m` (3.6). For each database of RIRs a file named `ASLP_final_exp5_Cm_`*DBname*`_...` contains the information needed to upload the results for the approximation of each RIR. The cell array `names`contains the names of the data where the results of the simulations performed with the script

`simulations.m` are stored. It is used in `showSimulResults.m` (3.7) for loading the data from the result files. It also contains the model complexity values C_m, the parameters used to build the pole grid (see `poleGrid.m`) and the number of unstable solutions that occurred for the BU method (for the OBF model) and the STMCB method (for the pole-zero model). Each result file (`ASLP_exp5_Cm_`$DBname_RIRname_\ldots$) contains the values of the Normalized Mean-Square Error (NMSE) computed in time and frequency domain, as described in [2]. They are loaded in `showPrecomputedResults.m`, where the means are computed and the results plotted as in Fig. 8-9 in [2].

4 DISCLAIMER

This program is free software: you can redistribute it and/or modify it 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.

This program is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General Public License for more details. You should have received a copy of the GNU General Public License along with this program. If not, see http://www.gnu.org/licenses/.

5 ACKNOWLEDGEMENTS

This research work was carried out at the ESAT Laboratory of KU Leuven, Department of Electrical Engineering (ESAT) (Kasteelpark Arenberg 10, Leuven B-3001), in the frame of

1. the FP7-PEOPLE Marie Curie Initial Training Network 'Dereverberation and Reverberation of Audio, Music, and Speech (DREAMS)' (http://www.dreams-itn.eu/), funded by the European Commission under Grant Agreement no. 316969,
2. KU Leuven Research Council CoE PFV/10/002 (OPTEC),
3. Research Project FWO no. G.0763.12 'Wireless Acoustic Sensor Networks for Extended Auditory Communication', and
4. was supported by a Postdoctoral Fellowship (F+/14/045) of the KU Leuven Research Fund.

The scientific responsibility is assumed by its authors.

REFERENCES

[1] G. Vairetti, T. van Waterschoot, M. Moonen, M. Catrysse, and S. H. Jensen, "An automatic model-building algorithm for sparse approximation of room impulse responses with orthonormal basis functions," in *Proc. Int. Workshop on Acoustic Signal Enhancement (IWAENC 2014)*, (Antibes-Juan Les Pins), 2014.

[2] G. Vairetti, E. De Sena, M. Catrysse, S. H. Jensen, M. Moonen, and T. van Waterschoot, "A scalable algorithm for physically motivated and sparse approximation of room impulse responses with orthonormal basis functions," tech. rep., KU Leuven, 2015.

[3] G. Vairetti, T. van Waterschoot, M. Moonen, M. Catrysse, and S. H. Jensen, "Sparse linear parametric modeling of room acoustics with orthonormal basis functions," in *Proc. 22nd Eur. Signal Process. Conf. (EUSIPCO'14)*, Lisbon, Portugal, 2014.

[4] T. Paatero and M. Karjalainen, "Kautz filters and generalized frequency resolution: Theory and audio applications," *J. Audio Eng. Soc.*, vol. 51, no. 1/2, pp. 27–44, 2003.

[5] H. Brandenstein and R. Unbehauen, "Least-squares approximation of FIR by IIR digital filters," *IEEE Trans. Signal Process.*, vol. 46, no. 1, pp. 21–30, 1998.

[6] J. Y. Wen, N. D. Gaubitch, E. A. Habets, T. Myatt, and P. A. Naylor, "Evaluation of speech dereverberation algorithms using the MARDY database," in *Proc. Int. Workshop on Acoustic Signal Enhancement (IWAENC 2006)*, (Paris, France), 2006. http://www.commsp.ee.ic.ac.uk/ sap/resources/mardy-multichannel-acoustic-reverberation-database-at-york-database/.

[7] J. K. Nielsen, J. R. Jensen, S. H. Jensen, and M. G. Christensen, "The single-and multichannel audio recordings database (SMARD)," in *Proc. Int. Workshop on Acoustic Signal Enhancement (IWAENC 2014)*, (Antibes-Juan Les Pins, France), pp. 40–44, 2014. http://www.smard.es.aau.dk/.

[8] E. Hadad, F. Heese, P. Vary, and S. Gannot, "Multichannel audio database in various acoustic environments," in *IEEE Proc. Int. Workshop on Acoustic Signal Enhancement (IWAENC 2014)*, (Antibes-Juan Les Pins), pp. 313–317, 2014. http://www.ind.rwth-aachen.de/en/research/tools-downloads/multichannel-impulse-response-database/.

[9] J. O. Smith and J. S. Abel, "Bark and ERB bilinear transforms," *IEEE Trans. Speech Audio Process.*, vol. 7, no. 6, pp. 697–708, 1999.

[10] A. Härmä, M. Karjalainen, L. Savioja, V. Välimäki, U. K. Laine, and J. Huopaniemi, "Frequency-warped signal processing for audio applications," *J. Audio Eng. Soc.*, vol. 48, no. 11, pp. 1011–1031, 2000.

© June 23, 2016