VPS VIBRATIONS IN PHYSICAL SYSTEMS

DOI: 10.21008/j.0860-6897.2022.2.02
Vibrations in Physical Systems, 2022, 33(2), 2022202

Enhancing Speech Signals Based on an MEMS Microphone Array and Temporal Differences in the Incoming Signal

Jan FELCYN[1], Michał RASZEWSKI[2]

[1] Department of Acoustics, Faculty of Physics, Adam Mickiewicz University,
[2] LARS Andrzej Szymański, Niepruszewo

Corresponding author: Jan FELCYN, email: janaku@amu.edu.pl

Abstract The development of the Internet of things and automatisation in everyday life also influences our houses. There are more and more devices on the market which can be controlled remotely. One kind of such control involves the use of voice signals. This method tends to use microphone arrays and dedicated algorithms to enhance the speech signal and recognize the words in it. In this project, a small 5-microphone array was developed. To enhance the quality of the signal, dedicated software was written. It consists of several modules, including the direction of arrival estimation, denoising, and differentiation between adults and children. The results showed that the custom algorithm can increase the signal to noise ratio by up to 6 dB.

Keywords: microphone array, speech enhancement, direction of arrival, signal processing.

1. Introduction

There are many devices on the market which can be controlled by voice commands. The voice assistants on smartphones are the most obvious examples. However, voice-controlled equipment can be much more sophisticated, including wheelchairs [1] and robots [2]. Nevertheless in many cases, a piece of hardware (like Amazon Echo or Google Home) and dedicated software are all that users need. Such a configuration allows smart homes to be controlled [3, 4] or elderly people to be assisted [5, 6] The simplest approach can be found in smartphones, where the signal is captured with only one microphone. However, when a user is far from the device, the incoming signal can be weak, with much reverberation, delay and other distortions. Thus, to enhance the speech signal some devices use microphone arrays (e.g. Amazon Echo).

Microphone arrays consist of multiple microphones, placed in different manners [7]. The simplest is a linear array, in which all the microphones are placed along one line. There are also 2D and 3D arrays. In the former, microphones are placed on the surface, while in 3D – on a sphere, cube or another solid figure [8]. More details about planning the geometry of microphone arrays can be found in [9, 10].

The easiest way to enhance speech signals using microphone arrays is with a delay-sum beamformer (DSB). The method is based on the fact that an acoustical wave incoming to multiple microphones has slightly different distances to pass. So the signal between microphones is delayed. The time of delay differs due to the distance between the microphones and the shape of the array. If we synchronize the signal to be in phase and sum it, the usable signal should be enhanced, while the noise – which we assume is random – should be decreased. This method is the easiest and – from point of view of programming – also the fastest. But it is limited by the length of the acoustical waves and the distances between microphones. Thus, several other approaches were also proposed. One of them is superdirective beamforming [11, 12]. When the crucial thing is the direction of arrival (DOA) estimation, generalized cross correlation (GCC) is used [13]. For microphone arrays which are parts of hearing aids, dereverberation is also an important operation [14].

In this project, the microphone array is only a small part of the bigger system. It is aimed at voice-control home devices, such as heaters, water valves, thermostats, furnaces, lights, fans, wall sockets, roller blinds, gates, pumps and so on. The overall flow of the signal is that a user utters a command. The signal comes to all 5 microphones, is summed properly to increase the signal-to-noise ratio, and is then sent to the modules of automatic speech recognition and the decision support system. Next the central unit sends commands to the peripheral devices of the Auraton Smart system (another system developed by LARS company) to perform user defined functions like switching on the lights or lowering the roller blinds. The prototype of the device is presented in Fig. 1.

Figure 1. The prototype of the main device.

As the system is intended to be fast, the whole computation process is done locally, using the middleware device. It is also safer, because no information is sent to cloud services or external companies – there are no issues regarding privacy which is sometimes a problem for the users of commonly used voice-control services [15, 16].

The software is based on customised Linux, works on an STM32MP157 microprocessor (ARM-A7 architecture) and runs two cores at 650 MHz. The platform is equipped with 1GB RAM, a Wi-Fi module as well as Ethernet and 868 MHz radio to communicate with peripheral devices. The prototype was also equipped with customised Python 3.6 installation. That is why Python was used while developing the algorithm.

Nowadays, Python is one of the most popular programming languages. There are also some acoustical packages for it, including for microphone arrays signal processing [17]. However, as we wanted to create a commercial product, we decided to write our own code to avoid possible legal conflicts and also to better control the whole procedure.

The main goal of our algorithm is to enhance speech using a simple DSB. Then, the enhanced signal goes to the speech recognition module. Thanks to that, many aspects of the house could be controlled using voice commands.

In next sections, we describe the microphone array we used, the code implementation (including different modules) and the performance of the whole system.

2. Methods

2.1. Microphone array and its initialization

Due to the limitations of the used components, the number of microphones was set at 5. During the projection stage we also decided that the array should be two-dimensional. Taking into account all these restrictions, a circular location of microphones was proposed. So the array is a circle with a diameter of 120 mm. MEMS microphones (model MP34DT05-A) were used and are placed regularly on the perimeter of one semicircle, on radiuses spaced by 45 degrees. The whole device also has a microprocessor and is connected with the main module via a micro-USB port. The prototype of array is presented in Fig. 2.

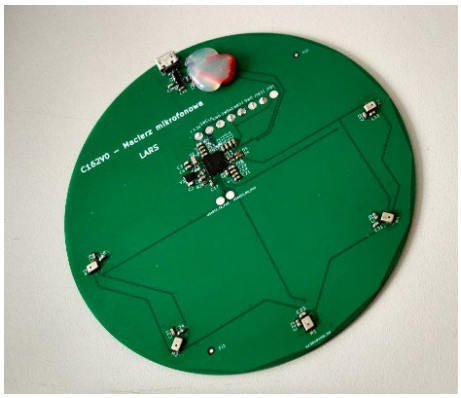

Figure 2. The prototype of the used microphone array.

When the array is connected with a computer, it is recognized as an audio interface. It enables five audio channels to be recorded at the same time. Each channel represents a different microphone. Looking at Fig. 2, from the left to the right side of the array the microphones are enumerated as follows: 3, 2, 1, 4, 5.

The key feature of the array is to sum signals from all microphones constructively. Thus, DOA estimation is needed and then a DSB algorithm is used. To accomplish this, first delay in samples between pairs of microphones has to be computed. Knowing the sampling frequency and geometry of the array (the distance between microphones), possible delay values can be computed in some angular resolution. In this case, we computed delay values by 10 degrees around the array, with the 0 angle being directly in front of microphone 1, in the same plane as the array. This angle was marked as Φ. Then, with the same angular resolution, the angle Θ (elevation) was added, which was in the vertical area; the angle directly above the centre of the array was 90 degrees. Having all this information, the matrix of delays was computed and was used as the reference to compare with the real-time incoming signal and estimate the DOA. In our code this function is called as matrix initialization.

2.2. Required software modules and their implementations

The main goal of using the microphone array in our project is to enhance a speech signal. To do this we decided to include several modules in the algorithm. The algorithm uses Python language and is written with the usage of two main files. The first file contains the definitions of all the classes and functions used in the programme, while the second script combines them together into different 'main' functions aimed at running code several times with different input parameters.

Each software module and its implementation is described below.

2.2.1. Voice Activity Detection (VAD)

This module is necessary as we do not want to analyse the incoming signal when it is not speech or when there is only background noise. The simplest way could be just setting up a threshold in decibels, however this would not allow other loud signals (which are not speech) to be excluded from the analysis. A more sophisticated way is to analyse the incoming signal in the spectral dimension. In order to do this, we included a simple creation of four mel filters in the algorithm. The incoming signal is filtered into four subbands and each subband is analysed in terms of its energy. The algorithm compares the maximum values of the signal with its mean amplitude to avoid a situation where constant noise would be analysed. There is also a threshold defined in the initialization of the array, which simply means that a signal quieter than the threshold would not be taken into account. This approach is of course very simple, but it was necessary to keep the algorithm as simple as possible.

2.2.2. Direction of arrival estimation

This module was tested in different approaches. The main idea was to use the GCC method, but it failed. The algorithm did not work properly, and the DOA estimation was impossible. We tested several methods and codes – based on different github repositories – but none of them was successful. Thus, a new idea was needed. We knew that the algorithm should be fast and functional in real-time. That is why more sophisticated methods could not be used. We decided to use the root mean square error (RMSE) function to find out which angle the signal was coming from.

The idea is simple: we know what the delays between microphones are, because we have already computed them in the initialization procedure. When the signal arrives at the array, we can identify the delays and compare them with the matrix of delays.

The procedure of DoA is as follows:

1. The incoming 5- channel signal is divided into 5 mono signals.
2. A smaller time window of each signal is selected – commonly half of the whole length – to speed up the process.
3. For each pair of signals one signal is time-shifted to the other sample by a sample in the defined range (e.g. -14 to +14) and for each combination RMSE is computed to express the similarity of signals. When the RMSE is the lowest, this delay is assumed to be the proper one.
4. This procedure is repeated ten times, as ten different pairs of signals can be defined (1-2, 1-3, 1-4, 1-5, 2-3, 2-4, 2-5, 3-4, 3-5, 4-5).
5. A vector of computed delays is compared with the delays from the initialisation matrix (again, RMSE is computed). The most similar vector is regarded as the proper one and its corresponding angle is assumed to be the angle from which the signal comes.

This approach is fast and gives reliable results, which will be presented in the appropriate section of this article.

2.2.3. Spectral de-noise

In this module several methods were investigated in terms of their simplicity and computational efficiency. We chose spectral subtraction, matrix decomposition based on covariance coefficients (subspace) and an iterative Wiener filter. Only the first method met the criteria of speed and was implemented.

Spectral subtraction was first proposed in 1979 [18], but nowadays there are many different approaches to it. In our algorithm we compute a filter gain coefficient based on the estimation of the signal and noise in a given time window. The used algorithm is based on the implementation of that filter in the Pyroomacoustics package for Python. However, to have better control, avoid legal issues and optimise the speed of the whole process, we rewrote the code to better suit our needs.

2.2.4. Signal interpolation

We did not plan to include this module in the algorithm, but was necessary due to the specific discontinuity sometimes encountered in the signal.

When the speaker is not moving around the array, the signal in each time frame is estimated to arrive from the same direction. However, when the speaker moves around the estimated angle for various time frames could be different. In some cases, due to the different angle estimation between neighbouring frames, some non-linearity occurs. Thus, the last sample from one frame and the first from another exhibit a big difference in amplitude and this causes clicks in the audio material. To avoid this, we included in the algorithm a simple linear interpolation which smooths the signal by adjusting the values of the samples in a given time window.

2.2.5. Moving average, lowpass and bandpass filters

As potential clicks are fast changes, it is necessary to lowpass the signal. Two ways of doing this were chosen: the simple implementation of Butterworth filters (both lowpass and bandpass) and moving average filter. As always, their cut-off frequencies have to be set responsibly, to keep all needed information in the signal. However, even with smooth performance, the results are audible.

2.2.6. Speaker recognition

Only adult people should be able to control the main device by voice. Thus, a simple algorithm was implemented to recognize the fundamental frequency of a voice. It is based on the autocorrelation approach and allows the threshold frequency to be set above that at which the voice would be recognized as that of a child.

3. The results of speech enhancement

The whole algorithm was developed step by step, thus its performance was rated several times, including different testing signals and circumstances. The crucial part was the DOA module, and it was tested with speech signals as well as testing ones (including pink noise and tone impulses).

According to [19], there is a simple equation which allows possible gain in SNR to be computed for a given microphone array. With 5 microphones, this gain should be around 7 dB. To check if this is true in reality, we conducted some measurements in an anechoic chamber.

The array was put on a loudspeaker stand, on the Ecophone material. One meter from it there was a loudspeaker which emitted testing signals. The array was rotated counter clockwise by 10 degrees and for each direction testing signals were recorded (35 different recordings were made). This procedure is presented in explanatory Fig. 3.

However, first we simply recorded the background noise and compared the amplitude of the signal from one microphone to the signal summed from all microphones. As we have mentioned earlier, the overall reduction of noise should be around 7 dB. The results of this comparison, done for all FFT subbands are presented in Fig. 4.

As one can see, the overall gain oscillates between 6 and 8 decibels and gets less stable with the increase in frequency. However, the overall mean reduction is 6.81 dB, thus very close to the prediction from Grythe's formula [19]. The next step was to analyse the array performance when the incoming signal is pink noise. The results of this comparison are presented in Fig. 5.

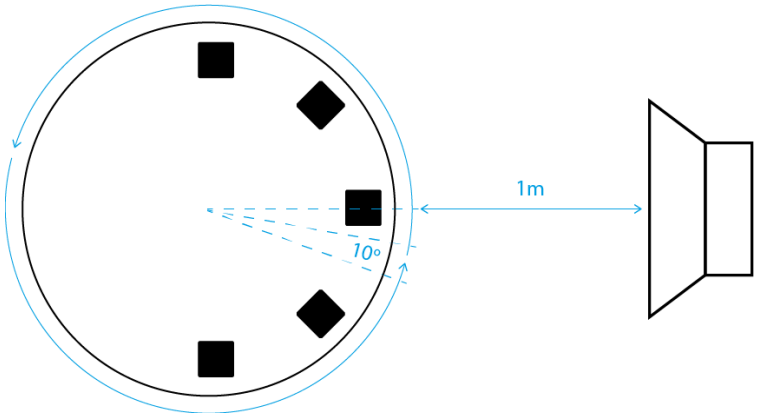

Figure 3. Explanatory figure presenting the procedure of recording testing signals in an anechoic chamber.

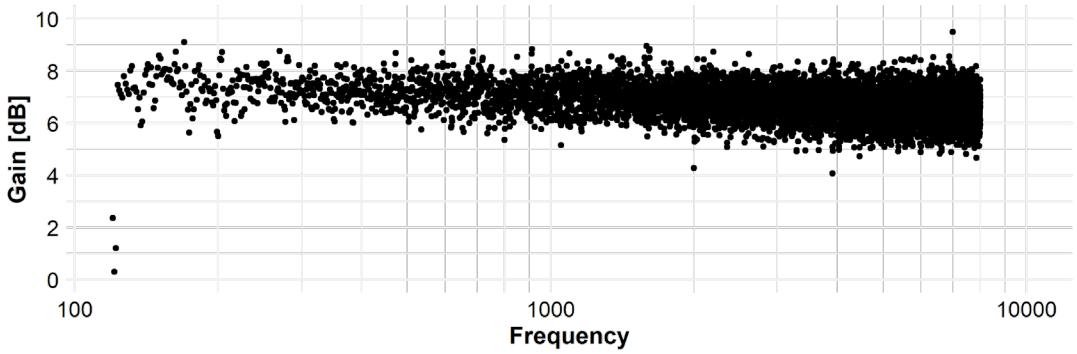

Figure 4. Gain in SNR computed by the comparison of the signal from a single microphone from the array with the signal summed from all 5 microphones; the input signal was only background noise.

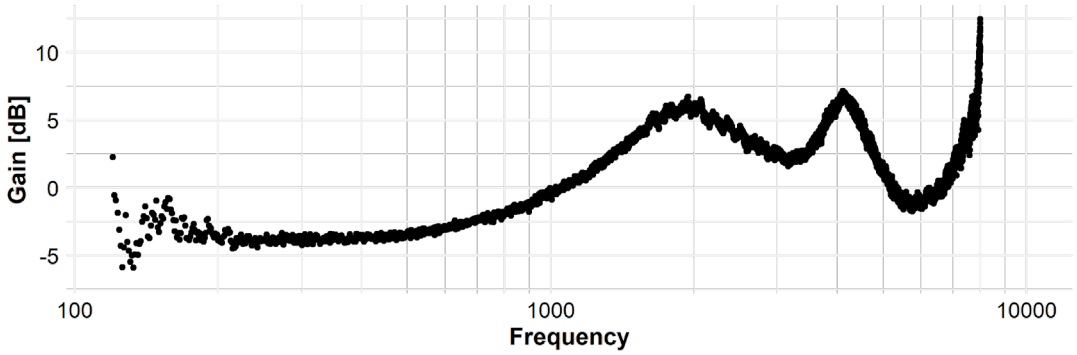

Figure 5. Gain in SNR computed by the comparison of the signal from a single microphone from the array with the signal summed from all 5 microphones; the input signal was pink noise.

Figure 6. Comparison of the signal with noise before (blue) and after (red) the spectral denoising algorithm. Change of SNR was around +5 dB.

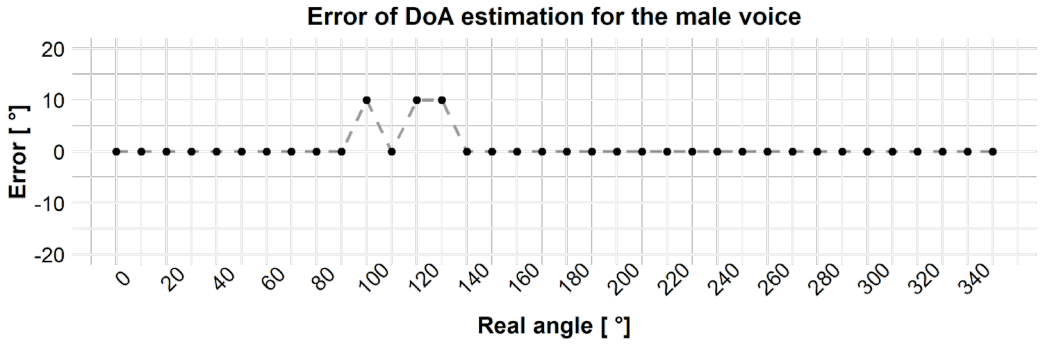

Figure 7. Error (in degrees) of estimation of direction of arrival for the microphone array – tested with a male voice.

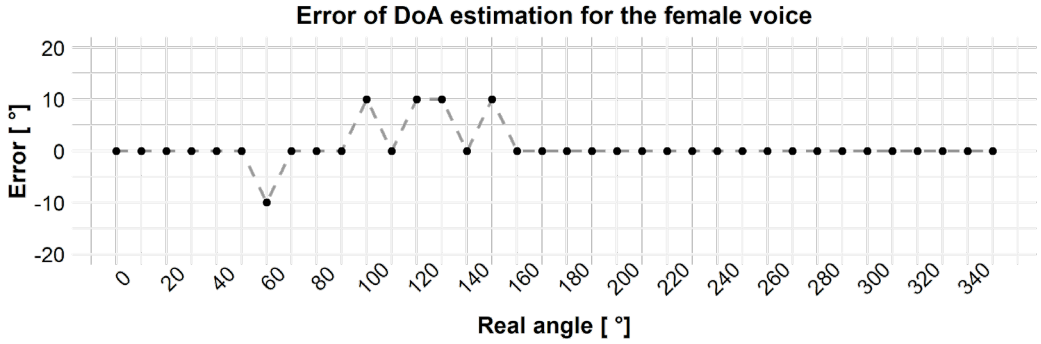

Figure 8. Error (in degrees) of estimation of direction of arrival for the microphone array – tested with a male voice.

This time the gain characteristics is not flat. Up to 1 kHz the gain is negative, meaning that the incoming frequencies are strengthened instead of weakened. The reason for this is the array geometry. The distances

between microphones are too small for low frequencies, thus they always sum up constructively. The peaks of the reduction are around 2 and 4 kHz, which is again the effect of the array's geometry. The overall gain this time is around 3 dB, which is of course far less than it was for the background noise, but is still satisfactory regarding the goals of the project.

We also checked the performance of the denoising algorithm. Several mixtures made of speech and pink noise were created, with different SNRs, from -6 to +12 dB (step by 3 dB). The results showed that in every case the denoising algorithm improved the SNR by at least 4 dB. An example of this improvement is shown in Fig. 6 (the original SNR was 0dB, the improvement was around 5 dB).

The last tests were conducted to rate the precision of the DOA algorithm. This was done by using both male and female speech samples – numbers in English. The loudspeaker was moved around the array by 10 degrees and the algorithm detected the DOA. The results are presented in Figs. 7 and 8.

As can be seen, for both male and female voices the error of estimated angle was not larger than 10 degrees. Please note that when the estimated angle was larger than it was in reality, the error is positive. In the other situation (the underestimation of a real angle), the error is negative.

4. Discussion and limitations

The main drawback of the chosen environment was that Python was not designed to be the fastest language. Python is a high-level programming language, and its main goal is clarity of a code. Thus, the code was improved multiple times to speed it up and make it as fast as possible.

One of the solutions was to use a package called Numba. This package allowed us to compile some functions to C language and use them in that form. This operation was a milestone in the code development. However, the number of functions supported by Numba is limited, so not every function could be compiled with it. Another way was to compare the performance of the same functions from different packages and choose the fastest. Thanks to all of the improvements, the code finally ran smoothly in real-time. Nevertheless, the spectral subtraction could not be included in the real-time version of the code, as it generated large delays. From today's point of view it seems that the best idea could be to use C++ with the JUCE framework – this is common for audio plug-ins in VST standard. However, it was not possible in this project.

All these limitations necessitated inventing a new approach to DoA estimation. As can be seen from figures above, the overall performance of the algorithm was still satisfactory and fulfilled the requirements established in the project at the outset. On its own, the array has limitations, mainly because of its small size – thus, low frequencies always sum each other up constructively, there are also some higher frequencies which amplify each other (see Fig. 4). However, by using a bandpass filter this phenomenon (for low frequencies) could be somehow limited.

5. Conclusions

The environment in which we worked had its limitations. Several problems, with both hardware and software, occurred during the development of the algorithm. However, the end results are satisfactory. The algorithm runs fast and has different additional modules which can be also used in other circumstances. The signal coming out of the algorithm is then sent to the automatic speech recognition module. The results of ASR are between 94% and 97%, which confirms that the signal is of high quality.

Acknowledgments

This project was financed by the National Centre for Research and Development with the grant POIR.01.01.01-00-0044/17. The spectral subtraction module in the algorithm was developed by Dawid Niemiec.

References

1. M.H. Jabardi; Voice controlled Smart Electric-Powered wheelchair based on Artificial Neural Network; International Journal of Advanced Research in Computer Science 2017, 8(5), 31–37. DOI: 10.26483/IJARCS.V8I5.3650
2. M.Y.A. Khan, H. Rasheed, U. Shahid; Voice Controlled Robot using Neural Network based Speech Recognition using Linear Predictive Coding; Bahria University Journal of Information & Communica-tion Technology 2016, 9, 47–49.
3. A. Brenon, F. Portet, M. Vacher; Arcades: A deep model for adaptive decision making in voice controlled smart-home; Pervasive and Mobile Computing 2018, 49, 92–110. DOI: 10.1016/j.pmcj.2018.06.011

4. A. Chandini, P.V. Bhaskar Reddy; Smart home automation using a voice-bot; International Journal of Advanced Research in Computer Science 2020, 11, 194–200. DOI: 10.26483/IJARCS.V11I0.6584
5. B. Busatlic, N. Dogru, I. Lera, E. Sukic; Smart homes with voice activated systems for disabled people; TEM Journal 2017, 6(1), 103–107. DOI: 10.18421/TEM61-15
6. K. O'Brien, A. Liggett, V. Ramirez-Zohfeld, P. Sunkara, L. A. Lindquist; Voice-Controlled Intelligent Personal Assistants to Support Aging in Place; J. Am. Geriatr. Soc. 2020, 68(1), 176–179. DOI: 10.1111/jgs.16217
7. G.W. Elko, J. Meyer; Microphone Arrays; In: Springer Handbook of Speech Processing; M. Brandstein, D. Ward, Eds.; Springer: Berlin, Heidelberg, 2008, 1021–1041.
8. D.P. Jarrett, E.A.P. Habets, P.A. Naylor; Theory and Applications of Spherical Microphone Array Processing; Springer: Cham, 2017.
9. J. Benesty, J. Chen; Study and Design of Differential Microphone Arrays; Springer: Berlin, 2013.
10. J. Chen, J. Benesty; Design and Implementation of Small Microphone Arrays for Acoustic and Speech Signal Processing, http://www.iwaenc2014.org/files/2014_IWAENC_MicrophoneArrays_Chen.pdf (Accessed: March 16, 2018).
11. S. Dodo, M. Moonen; Superdirective beamforming robust against microphone mismatch; In: 2006 IEEE International Conference on Acoustics, Speech and Signal Processing – Proceedings; Toulouse, France, May 14-19, 2006; IEEE: Piscataway, USA, 2006, Vol. 5.
12. G. Huang, J. Benesty, J. Chen; Subspace superdirective beamforming with uniform circular microphone arrays; Proceedings of 2016 International Workshop on Acoustic Signal Enhancement (IWAENC); Xi'an, China, September 13-16, 2016; IEEE: Piscataway, USA, 2016.
13. B. van den Broeck, A. Bertrand, P. Karsmakers, B. Vanrumste, H. van Hamme, M. Moonen; Time-domain GCC-phat sound source localization for small microphone arrays; Proceedings of 5th European DSP Education and Research Conference (EDERC); Amsterdam, Netherlands, September 13-14, 2012; IEEE: Piscataway, USA, 2012, 76-80. DOI: 10.1109/EDERC.2012.6532229
14. A. Kuklasinski; Multi-channel dereverberation for speech intelligibility improvement in hearing aid applications; Ph.D. Thesis, Aalborg University, Aalborg, Denmark, 2016.
15. L. Hernández Acosta, D. Reinhardt; A survey on privacy issues and solutions for Voice-controlled Digital Assistants; Pervasive and Mobile Computing 2022, 80, 101523. DOI: 10.1016/j.pmcj.2021.101523
16. R. Hasan, R. Shams, M. Rahman; Consumer trust and perceived risk for voice-controlled artificial intelligence: The case of Siri; Journal of Business Research 2021, 131, 591–597. DOI: 10.1016/j.jbusres.2020.12.012
17. E. Sarradj, G. Herold; A Python framework for microphone array data processing; Applied Acoustics 2017, 116, 50–58. DOI: 10.1016/J.APACOUST.2016.09.015
18. M. Berouti, R. Schwartz, J. Makhoul; Enhancement of speech corrupted by acoustic noise; In: ICASSP'79. IEEE International Conference on Acoustics, Speech, and Signal Processing; Washington, USA, April 2-4, 1979; IEEE: Piscataway, USA, 1979, 208–211. DOI: 10.1109/icassp.1979.1170788
19. J. Grythe; Array gain and reduction of self-noise; Norsonic, Technical note, Oslo, 2016.