MICARRAYLIB: SOFTWARE FOR REPRODUCIBLE AGGREGATION, STANDARDIZATION, AND SIGNAL PROCESSING OF MICROPHONE ARRAY DATASETS

Iran R. Roman[1*], *Juan Pablo Bello*[1,2]

[1] Music and Audio Research Lab, New York University, NY, USA
[2] Center for Urban Science and Progress, New York University, NY, USA

ABSTRACT

`micarraylib` is a python library to load, standardize, and aggregate datasets collected with different microphone array hardware. The goal is to create larger datasets by aggregating existing and mostly incompatible microphone array data and encoding it into standard B-format ambisonics. These larger datasets can be used to develop novel sound event localization and detection (SELD) algorithms. `micarraylib` streamlines the download, load, resampling, aggregation, and signal processing of datasets collected with commonly-used and custom microphone array hardware. We provide an API to standardize the 3D coordinates of each microphone array capsule, visualize the placement of microphone arrays in specific spatial configurations, and encode time-series data collected with different microphone arrays into B-format ambisonics. Finally, we also show that the data aggregates can be used to reconstruct a microphone capsule's time-series data using the information from other capsules in the data aggregate. `micarraylib` will allow for the easy addition of more datasets and microphone array hardware as they become available in the future. All original software written for this paper is released with an open-source license.

***Index Terms*—** sound event detection and localization, microphone arrays, spherical harmonics, ambisonics encoder, multichannel signal processing

1. INTRODUCTION

In the past few years, the field of machine listening has seen major advances in sound event detection (SED) algorithms [1, 2, 3]. These advances have been made possible by the introduction of large datasets with annotated sound events. In particular, the millions of annotated soundclips in AudioSet [4], totaling around 100 hours of data, have been critical for these developments.

In contrast, the development of sound event localization and detection (SELD) algorithms has been slower. This is not surprising, given that SELD datasets are much smaller than AudioSet, usually with only a few thousand sound events with annotations for both category and spatial localization (example datasets include those introduced by the DCASE SELD challenges in 2019 [5], 2020 [6], and 2021 [7], as well as the LOCATA challenge [8]).

To develop SELD algorithms that are as robust as existing SED ones, machine listening researchers will need access to large amounts of data collected with microphone arrays. A number of publicly available microphone array datasets exist (see Table 1), but these datasets have heterogeneous hardware parameters, thus complicating their aggregation.

*roman@nyu.edu

Table 1: Some publicly available microphone array datasets. The number of microphone arrays, total microphone capsules, length in hours, and presence of SELD annotations are tabulated.

dataset	no. arr	capsules	length	SELD
DCASE(3) 2019 [5]	1	4	8 Hr	Yes
DCASE(3) 2020 [6]	1	4	13 Hr	Yes
DCASE(3) 2021 [7]	1	4	13 Hr	Yes
LOCATA [8]	4	63	0.5 Hr	Yes
3D-MARCo [9]	7	71	0.2 Hr	No
EigenScape [10]	1	32	11 Hr	No

Two notorious differences between microphone array datasets include 1) the use of different microphone hardware and 2) conventions for SELD annotation (or complete lack of), including event start time, duration, and position in space. To standardize microphone array recordings across different hardware, some researchers encode them into the ambisonics B-format [11, 12], which uses the individual microphone capsule coordinates to compute a matrix of spherical harmonic coefficients. In its simplest form, the B-format is obtained by multiplying the pseudo-inverse of this matrix by the corresponding raw capsule recordings (also known as A-format) [13]. The ambisonics B-format captures specific spatial features (i.e. the first channel is equivalent to an omnidiectional microphone, the next three channels are fig-8 microphones aligned on the x, y, and z Cartesian coordinates, etc. see [14] for details on the spherical harmonics theory that results in B-format encoding). On the other hand, standardizing SELD annotations is possible if certain parameters (i.e. event start time, end time, and a position in space) are parsed to be consistent across datasets. Additionally, since not all microphone array datasets have SELD annotations, the use of unsupervised and self-supervised learning strategies to learn spatial representations will be necessary to use all available datasets.

Here we introduce `micarraylib`, a python library to download, standardize, and aggregate existing microphone array recordings. Using `micarraylib`, one can encode raw microphone array recordings across different datasets to be in the common ambisonics B-format. `micarraylib` also standardizes annotations to be in a common convention. Additionally, `micarraylib` organizes metadata (i.e. microphone capsule coordinates and hardware name) to be readily accessible. `micarraylib` is freely available at `https://github.com/micarraylib/micarraylib`.

In the next sections we describe `micarraylib`'s functional principles and show example applications, which include the aggregation of different SELD datasets, visualization of aggregated microphone coordinates, and data augmentation via interpolation of a virtual capsule recording using data from neighboring capsules.

2. LIBRARY FUNCTION

We standardize three elements present in most microphone array datasets: 1) metadata, 2) SELD labels (if any), and 3) audio format.

2.1. Metadata processing

The most important piece of metadata accompanying any microphone array dataset is its microphone capsule coordinates. Microphone array datasets often use different microphone hardware. Designers publish the relative distance between microphone capsules and a reference point. These can be converted to 3D coordinates (Cartesian or polar). In some datasets the reference point is the microphone array's center, or a specific location in a physical space. `micarraylib` standardizes these distances and locations to be a common coordinate format (3D vectors, either Cartesian or polar). These coordinates can be used to visualize microphone arrays, to compute spherical harmonics, and more generally to develop algorithms that incorporate spatial information at the level of microphone capsule location.

Other pieces of metadata that `micarraylib` processes (if available) and makes available to the user include sound scene category, musical artist, and geographic location of the recording.

2.2. SELD label standardization

Because of their unique ability to capture spatial information, microphone arrays are a common hardware choice to collect SELD data. While several labeling conventions exist among datasets, annotations for sound events include at least a start time, but may also have an end time, and a position in space. `micarraylib` standardizes sound event labels across datasets to have the following format: start and end time (python tuple), object category (a unique integer or string), location coordinates (3D vector that may change over a fixed time-step if the event moves), and active time-steps (list of booleans if the event is transiently on or off). When one of these parameters does not exist for an event, `micarraylib` will indicate it with a `None`. By using `micarraylib`, researchers will access microphone array datasets with a standard format for spatial sound event labels.

2.3. Audio standardization

The ambisonics B-format allows for the standardization of microphone array recordings [11, 12]. A simple ambisonics encoder uses the individual microphone capsule coordinates to compute a matrix of spherical harmonics. The pseudo-inverse of this matrix then multiplies the raw capsule recordings to encode them into B-format channels.

While more complicated encoders are usually proprietary and include multiple effects that aid perceptual parameters [15], computing the spherical harmonics using microphone array coordinates is a straightforward operation if the microphone capsule coordinates are known. The Nth-order spherical harmonic matrix can be computed using equation 1.

$$Y_{n,l}(\theta,\phi) = X_{n,|l|}P_{n,|l|}\cos(\theta)\begin{cases}\sqrt{2}\sin(|l|\phi) & \text{if } l<0\\ 1 & \text{if } l=0\\ \sqrt{2}\cos(l\phi) & \text{if } l>0\end{cases} \quad (1)$$

This is the conventional equation used in the field of acoustics to compute the Laplace spherical harmonics [16]. n indexes the spherical harmonic order (i.e. 0th, 1st, 2nd, ..., Nth) and m indexes the degree (i.e. each order n has degrees $m \in [-n, n]$ degrees). $\theta \in [0, \pi]$ is the vertical angle advancing from top to bottom, $\phi \in [0, 2\pi]$ is the azimuth angle starting at the front of the microphone array and advancing counter-clockwise. $X_{n,|m|}$ is a normalization factor that ensures that spherical harmonics have unit magnitude [16] and $P_{n,|m|}$ is the Legendre function (without the Condon-Shortley phase) [17].

`micarraylib` converts raw capsule recordings into the common ambisonics B-format. This results in audio signals that have shared spatial characteristics across channels, independent of which microphone hardware was used to collect them.

3. LIBRARY ORGANIZATION AND COPYRIGHT

`micarraylib` is written in python and all its contents are open-source. The following subsections describe the organization of its file structure as seen from its root directory.

3.1. Micarrays

Directory that contains the `array_shapes_raw.py` and `array_directions_raw.py` files, which list the raw (as released by the manufacturer) shape (i.e. coordinates) and capsule directionality of each microphone array supported. The names that the manufacturer gave to each capsule in the microphone array are also included in these files. The `micarray.py` file in this directory defines a `micarray` object with attributes that summarize all the information provided by the manufacturer.

This directory also has files with functions that process the data from the `array_shapes_raw.py` and `array_directions_raw.py` files and standardize it to be in 3D Cartesian and/or polar coordinates.

3.2. Util

Directory that contains a `utils.py` file with basic functionalities for dataset standardization, such as functions to convert between polar and Cartesian coordinates, normalize units of length to meters and radians, and normalize time units to seconds (in SELD labels, for example). It also contains a `plotting.py` file with functions that tailor `matplotlib`'s plotting for microphone arrays.

3.3. Encoder

File defining the `encoder` object with its main attribute being a set of capsule coordinates (used to calculate the matrix of spherical harmonics). It also has an `encode` method that takes a numpy array with raw recordings and returns a simple encoding of these recordings in ambisonics B-format.

3.4. Dataset

File that defines the `dataset` object using the `soundata` API [18]. `soundata` is a new python library with tools to download and load common audio datasets with corresponding annotations and metadata. In addition to `soundata` attributes, the `dataset` object includes a list of microphone capsule coordinates used. The `soundata` API includes all methods to download and load the original data. micarraylib standardizes the SELD annotations when they are not standardized by soundata.

3.5. Aggregator

File that defines the aggregate_datasets object, whose attributes include a list of dataset objects. Its default method standardizes all recordings across datasets to be the same number of channels in ambisonics B-format, and pairs individual recordings with their corresponding SELD labels.

It also defines the micarray_aggregate object, which aggregates coordinates and recordings across microphone arrays (useful when multiple pieces of microphone array hardware are used together in a single dataset, such as the 3D-MARCo or LOCATA datasets).

3.6. Augmentation

File that defines a data_augmentation object, which uses a neural network model to virtually add capsule recording data to a microphone array dataset at a coordinate defined by the user. Section 5 below describes the current functionality of this model (which is limited to reconstruction of channels within the EigenMike [19] hardware at the time of this writing, but we are working to expand its possibilities).

3.7. Copyright

micarraylib is released with a Creative Commons License. We also do not alter any dataset's license, as micarraylib only accesses data already hosted online (via soundata; as a result, datasets are not redistributed by micarraylib).

4. AGGREGATING DATASETS

micarraylib streamlines the aggregation of existing microphone array datasets. Figure 1 shows the code needed to standardize and aggregate the six different datasets in Table 1.

One at a time, micarraylib separately encodes each recording into a first-order ambisonics B-format (4 channels total; the first-order ambisonics limit is determined by the dataset with the lowest number of raw capsule recordings: 4 channels in the DCASE SELD datasets). After the simple encoding step, we have a total of 46 hours of audio data in a common ambisonics B-format. micarraylib also standardizes the SELD labels from the DCASE SELD and LOCATA datasets to have a start and end time, object category, spatial coordinates, and active time-steps. The 3D-MARCo and EigenScape datasets do not have SELD labels, and the resulting aggregate indicates this with None entries in the label attribute for those specific recordings. In the end, 34 hour of data in this dataset aggregate have labeled sound events.

4.1. Hardware considerations and next steps

Aggregating different datasets can result in SELD methods that confound elements that are different between datasets (i.e. hardware, events, and/or ambient). While our library encodes all datasets into the standard ambisonics B-format, it is important to keep in mind that the hardware differences between datasets could remain in the B-format. For this reason, we plan to continue fine-tuning our encoder to quantify and reduce differences between hardware. To better quantify these effects, datasets with scenes and events that

```python
import micarraylib as mc

datadir = '~/datasets'

datasets = [
    mc.datasets.dcase19(datadir),
    mc.datasets.dcase20(datadir),
    mc.datasets.dcase21(datadir),
    mc.datasets.locata(datadir),
    mc.datasets.marco(datadir),
    mc.datasets.eigenscape(datadir)
]

for dataset in datasets:
    dataset.load() # using the soundata API [18]

aggregate = mc.aggregators.aggregate_datasets(
    datasets,
    sr=24000,
)
```

Figure 1: Downloading, loading, and aggregating the six datasets in Table 1 using micarraylib.
.

are simultaneously collected with different hardware (i.e. the 3D-MARCo and LOCATA datasets) will be particularly useful.

5. DATA AUGMENTATION

An idealized spatial recording of a sound scene would record information at all locations in the space continuum. Since such idealized scenario is not possible with existing hardware, researchers must sample specific locations using microphone arrays with capsules at specific coordinates. micarraylib includes a model can be used to virtually add microphones to a dataset via interpolation from existing microphone capsule data.

5.1. Technical motivation

Aggregating microphone arrays can lead to denser spatial sampling of a sound scene. The resulting dense samplings of a sound scene are redundant [20]. Therefore, given a set of microphone capsules recording a common scene or source, it should be possible to interpolate, with some error, one of the capsule's time-series using the recordings collected with all other capsules. If this is possible, it should also be possible to virtually generate the recording of a microphone capsule outside but near the microphone array topology.

While a detailed empirical study of virtual microphone capsule time-series generation deserves a separate scientific report, micarraylib already includes some of this functionality. Here we describe a series of experiments that we carried out to design a model able to virtually add the recording of a missing microphone capsule using the recordings from other capsules in the EigenMike microphone array. These experiments also show the utility of micarraylib to aggregate datasets that can then be used for machine listening research.

In all experiments we ask the question: can the recording of a microphone capsule be reconstructed given the preceding 5 milliseconds of recordings with neighboring capsules? We hypothesize that such reconstruction is possible using both the recordings from neighboring capsules and their 3D spatial coordinates.

Table 2: Model performance for capsule interpolation experiments

Model	MSE (eval)
before training	0.9
exp 1	0.000039
exp 2	0.00013
exp 3	0.0016

5.2. Data

We used `micarraylib` to aggregate the EigenMike data from the EigenScape, 3D-MARCo, and LOCATA datasets. We skip the encoding step and keep the recordings in the raw format (A-format). We split each recording into development (the first 80% frames) and testing (the last 10% frames) subsets (the frames between the %80 and 90% of each recording length were left out to minimize the effects of temporal correlation between the development and testing data).

5.3. Methodology

The input features to train our model are an intermediate step between A-format and B-format that is computed as follows. We calculate the fourth order spherical harmonic matrix using the 3D coordinates for each EigenMike capsule and equation (1). The result is a matrix W that has $(N+1)^2$ rows ($N = 4$) and 32 columns, one corresponding to each EigenMike capsule. We multiply the sample recorded by a given capsule with the spherical harmonic coefficients in the corresponding column of matrix W. We stack 20 of these matrices into a tensor that corresponds to 5 consecutive milliseconds of samples in a recording. We refer to these concatenated matrices as tensor X.

In the experiments described below we remove some audio information from tensor X and we train a simple LeNet CNN [21] to use X to reconstruct the recording of a single capsule (whose audio information is missing from X) with a MSE loss function.

5.4. Experiments and results

We carried out three experiments: 1) Tensor X is only missing the audio information of the single capsule that we want to reconstruct. 2) The samples of five neighboring capsules are also removed from X. 3) X only has the audio information of the three capsules that would result in a tetrahedral geometry (4 capsules total) with respect to the capsule whose recording we want to reconstruct.

We trained a LeNet CNN model from scratch using the ADAM optimizer with a patience criterion of 1000 epochs. For each of these three experiments, the model's task was to reconstruct the missing samples of a single microphone capsule in the EigenMike using X as input. The average MSE across datapoints in the evaluation set was computed after training and is shown in Table 2.

As shown in Table 2, the evaluation MSE before training was close to 1. Experiment 1 reduced this MSE by 5 orders of magnitude, showing that a single capsule's samples can be reconstructed (with some error) using all other 31 capsule recordings in the EigenMike. Experiment 2 also reduced the MSE but only by 4 orders of magnitude, showing that reconstruction of the same capsule's recording is affected by the fact that audio information from neighboring capsules was missing. Experiment 3 only reduced the MSE by 3 orders of magnitude due to there only being information from

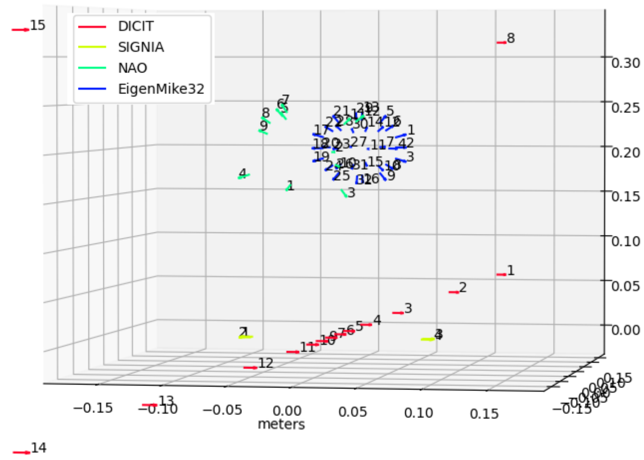

Figure 2: Interactive visualization with `micarraylib` of the location and directionality of microphone arrays used to collect the LOCATA dataset, one of which is the EigenMike. Using `matplotlib`'s 3D plotting, `micarraylib` includes methods to plot capsules in microphone array hardware.

3 other microphone capsules to reconstruct the capsule's recording. These results show that reconstructing a capsule's samples using other capsule's recordings is possible, but the interpolation is sensitive to the density and proximity of recordings available to carry out this reconstruction.

5.5. Next steps

Given that reconstructing a microphone's signal seems possible within EigenMike capsules, we will train a model able to use EigenMike data to recover the signal captured by a capsule or array outside the EigenMike geometry. For this purpose, we will use the microphone array aggregates from the 3D-MARCo and LOCATA datasets, which include data recorded by microphone capsules outside the EigenMike structure but very near to it (see Fig 2). Our goal is for `micarraylib` to ultimately allow for the virtual simulation of microphone capsules in arbitrary coordinates.

The idea of virtual microphone capsule generation has already been explored with supervised learning using neural network techniques like CNNs [22, 23] or autoencoders [24], via statistical interpolation with β-divergence [25] or with pure signal processing [26]. The full potential of the microphone array data augmentation that we propose will be achieved by using `micarraylib` in combination with audio augmentation libraries like MUDA [27] and Audiomentations (`https://github.com/iver56/audiomentations`).

6. CONCLUSIONS AND FUTURE DIRECTIONS

We have presented `micarraylib`, a library to aggregate microphone array datasets by standardizing microphone array coordinates, SELD labels, and recordings using a simple ambisonics B-format encoder. Our presentation of the library also included practical demonstrations that show both the utility of `micarraylib` and the need for it in the field of machine listening. Researchers will be able to use `micarraylib` to create large aggregates of standardized microphone array datasets.

We are also looking forward to seeing what other functionalities the machine listening research community believes should be added to `micarraylib`. We will continue adding new datasets to `micarraylib` in years to come, and we will also add support for more complex signal processing procedures that include completely custom array shapes, microphone polarity, methods for perceptually-plausible ambisonics encoding and decoding, as well as processing of motion-capture data for moving sound events and microphone arrays.

7. ACKNOWLEDGMENT

This material is based upon work supported by the National Science Foundation under grant no. IIS-1955357. The authors thank the founding source and their grant collaborators.

8. REFERENCES

[1] Y. Gong, Y.-A. Chung, and J. Glass, "AST: Audio spectrogram transformer," *arXiv preprint arXiv:2104.01778*, 2021.

[2] A. Jaegle, F. Gimeno, A. Brock, A. Zisserman, O. Vinyals, and J. Carreira, "Perceiver: General perception with iterative attention," *arXiv preprint arXiv:2103.03206*, 2021.

[3] A. Jansen, M. Plakal, R. Pandya, D. P. Ellis, S. Hershey, J. Liu, R. C. Moore, and R. A. Saurous, "Unsupervised learning of semantic audio representations," in *2018 IEEE international conference on acoustics, speech and signal processing (ICASSP)*. IEEE, 2018, pp. 126–130.

[4] J. F. Gemmeke, D. P. Ellis, D. Freedman, A. Jansen, W. Lawrence, R. C. Moore, M. Plakal, and M. Ritter, "Audio set: An ontology and human-labeled dataset for audio events," in *2017 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2017, pp. 776–780.

[5] S. Adavanne, A. Politis, and T. Virtanen, "A multi-room reverberant dataset for sound event localization and uetection," in *Submitted to Detection and Classification of Acoustic Scenes and Events 2019 Workshop (DCASE2019)*, 2019. [Online]. Available: https://arxiv.org/abs/1905.08546

[6] A. Politis, S. Adavanne, and T. Virtanen, "A dataset of reverberant spatial sound scenes with moving sources for sound event localization and detection," in *Proceedings of the Workshop on Detection and Classification of Acoustic Scenes and Events (DCASE2020)*, November 2020. [Online]. Available: https://arxiv.org/abs/2006.01919

[7] A. Politis, S. Adavanne, D. Krause, A. Deleforge, P. Srivastava, and T. Virtanen, "A dataset of dynamic reverberant sound scenes with directional interferers for sound event localization and detection," *arXiv preprint arXiv:2106.06999*, 2021. [Online]. Available: https://arxiv.org/abs/2106.06999

[8] H. W. Löllmann, C. Evers, A. Schmidt, H. Mellmann, H. Barfuss, P. A. Naylor, and W. Kellermann, "The LOCATA challenge data corpus for acoustic source localization and tracking," in *2018 IEEE 10th Sensor Array and Multichannel Signal Processing Workshop (SAM)*. IEEE, 2018, pp. 410–414.

[9] H. Lee and D. Johnson, "An open-access database of 3D microphone array recordings," in *Audio Engineering Society Convention 147*. Audio Engineering Society, 2019.

[10] M. C. Green and D. Murphy, "EigenScape: A database of spatial acoustic scene recordings," *Applied Sciences*, vol. 7, no. 11, p. 1204, 2017.

[11] J. Bamford and J. Vanderkooy, "Ambisonic sound for us (an analysis of imaging in ambisonics, stereo and dolby surround systems)," in *99th AES Convention*, 1995, pp. 6–9.

[12] F. Zotter, H. Pomberger, and M. Frank, "An alternative ambisonics formulation: Modal source strength matching and the effect of spatial aliasing," in *Audio Engineering Society Convention 126*. Audio Engineering Society, 2009.

[13] L. McCormack, S. Delikaris-Manias, A. Farina, D. Pinardi, and V. Pulkki, "Real-time conversion of sensor array signals into spherical harmonic signals with applications to spatially localized sub-band sound-field analysis," in *Audio Engineering Society Convention 144*. Audio Engineering Society, 2018.

[14] K. Atkinson and W. Han, *Spherical harmonics and approximations on the unit sphere: an introduction*. Springer Science & Business Media, 2012, vol. 2044.

[15] M. Frank, F. Zotter, and A. Sontacchi, "Producing 3D audio in ambisonics," in *Audio Engineering Society Conference: 57th International Conference: The Future of Audio Entertainment Technology–Cinema, Television and the Internet*. Audio Engineering Society, 2015.

[16] E. G. Williams, *Fourier acoustics: sound radiation and nearfield acoustical holography*. Academic press, 1999.

[17] M. Abramowitz and I. A. Stegun, *Handbook of mathematical functions with formulas, graphs, and mathematical tables*. US Government printing office, 1948, vol. 55.

[18] M. Fuentes, J. Salamon, P. Zinemanas, M. Rocamora, G. Plaja, I. R. Román, R. Bittner, M. Miron, X. Serra, and J. P. Bello, "Soundata: A Python library for reproducible use of audio datasets," *http://arxiv.org/abs/2109.12690*, 2021.

[19] M. Acoustics, "EM32 Eigenmike microphone array release notes (v17. 0)," *25 Summit Ave, Summit, NJ 07901, USA*, 2013.

[20] Y. Huang, T. Z. Shabestary, A. Gruenstein, and L. Wan, "Multi-microphone adaptive noise cancellation for robust hotword detection," 2019.

[21] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, 1998.

[22] T. Ochiai, M. Delcroix, T. Nakatani, R. Ikeshita, K. Kinoshita, and S. Araki, "Neural network-based virtual microphone estimator," in *ICASSP 2021-2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2021, pp. 6114–6118.

[23] K. Yamaoka, L. Li, N. Ono, S. Makino, and T. Yamada, "CNN-based virtual microphone signal estimation for MPDR beamforming in underdetermined situations," in *2019 27th European Signal Processing Conference (EUSIPCO)*. IEEE, 2019, pp. 1–5.

[24] R. Takahashi, L. Li, S. Makino, and T. Yamada, "VMInNet: Interpolation of virtual microphones in optimal latent space explored by autoencoder."

[25] H. Katahira, N. Ono, S. Miyabe, T. Yamada, and S. Makino, "Generalized amplitude interpolation by β-divergence for virtual microphone array," in *2014 14th International Workshop on Acoustic Signal Enhancement (IWAENC)*. IEEE, 2014, pp. 149–153.

[26] H. Katahira, N. Ono, S. Miyabe, T. Yamada, and S. Makino, "Virtually increasing microphone array elements by interpolation in complex-logarithmic domain," in *21st European Signal Processing Conference (EUSIPCO 2013)*. IEEE, 2013, pp. 1–5.

[27] B. McFee, E. J. Humphrey, and J. P. Bello, "A software framework for musical data augmentation." in *ISMIR*, vol. 2015, 2015, pp. 248–254.