Audio Engineering Society

Convention Express Paper 319

Presented at the AES 158th Convention
2025 May 22–24, Warsaw, Poland

This Express Paper was selected on the basis of a submitted synopsis that has been peer-reviewed by at least two qualified anonymous reviewers. The complete manuscript was not peer reviewed. This Express Paper has been reproduced from the author's advance manuscript without editing, corrections, or consideration by the Review Board. The AES takes no responsibility for the contents. This paper is available in the AES E-Library (http://www.aes.org/e-lib), all rights reserved. Reproduction of this paper, or any portion thereof, is not permitted without direct permission from the Journal of the Audio Engineering Society.

Binamix — A Python Library for Generating Binaural Audio Datasets

Dan Barry[1], Davoud Shariat Panah[1], Alessandro Ragano[1], Jan Skoglund[2], and Andrew Hines[1]

[1]*School of Computer Science, University College Dublin, Dublin, Ireland*
[2]*Google LLC, San Francisco, CA*

Correspondence should be addressed to Dan Barry (dan.barry@duck.com)

ABSTRACT

The increasing demand for spatial audio in applications such as virtual reality, immersive media, and spatial audio research necessitates robust solutions to generate binaural audio data sets for use in testing and validation. Binamix is an open-source Python library designed to facilitate programmatic binaural mixing using the extensive SADIE II Database, which provides Head Related Impulse Response (HRIR) and Binaural Room Impulse Response (BRIR) data for 20 subjects. In this paper, we present Binamix, an open-source library which provides a flexible and repeatable framework for creating large-scale spatial audio datasets, making it an invaluable resource for codec evaluation, audio quality metric development, and machine learning model training. A range of pre-built example scripts, utility functions, and visualization plots further streamline the process of custom pipeline creation. This paper presents an overview of the library's capabilities, including binaural rendering, impulse response interpolation, and multi-track mixing for various speaker layouts. The tools utilize a modified Delaunay triangulation technique to achieve accurate HRIR/BRIR interpolation where desired angles are not present in the data. By supporting a wide range of parameters such as azimuth, elevation, subject Impulse Responses (IRs), speaker layouts, mixing controls, and more, the library enables researchers to create large binaural datasets for any downstream purpose. We release the library under the Apache 2.0 License at https://github.com/QxLabIreland/Binamix/

1 Introduction

Spatial audio has seen a significant rise in popularity across various domains in recent years, from virtual reality and immersive media to auditory research and perceptual evaluation. The release of new audio formats such as Dolby Atmos [1] and Google Eclipsa [2] provide the necessary platforms on which the modern era of spatial audio is being built. A significant rise in media consumption through headphones over the last decade [3] is also contributing to advances in the binaural rendering technology required to deliver spatial audio over headphones. As research in the areas

of spatial audio codec creation and objective quality metrics [4] grows, the need for large-scale spatial audio data creation also increases. Such data is essential for testing and validation in codec and quality metric design. Additionally, large spatial audio datasets can be used to train machine learning algorithms for tasks like machine listening and no-reference quality estimation. Although existing tools support interfacing with HRIR libraries [5] and binaural rendering [6], no solution combines them with robust binaural mixing tools, surround speaker layout emulation, and dataset generation.

2 Binamix

Here, we present Binamix [7], an open-source Python 3 library for binaural audio dataset generation. The library contains a set of pipeline tools for programmatic binaural mixing, leveraging the extensive SADIE II Database [7] which contains high-resolution HRIR and BRIR data for 20 subjects. HRIRs or BRIRs are necessary for realistic binaural rendering. SADIE II contains IRs for two dummy head subjects with 8802 IR sample points around the sphere and 18 human subjects with up to 2818 IR sample points around the sphere for the HRIRs and a more sparse sphere sampling for the BRIRs. SADIE was chosen due to its widespread use in both research and commercial applications but Binamix can be used with other HRIR datasets by providing your own scripts to interface with the IR files.

2.1 Motivation

Our main work is principally focused on building audio quality metrics [4][8][9][10] and as such we require large amounts of labeled data for testing and training. Datasets do exist for testing traditional audio quality metrics [11] but no such binaural audio dataset exists for testing spatial quality metrics. It is impractical to create enough data manually using spatial audio production tools in a Digital Audio Workstation (DAW), and furthermore, generating all of the test condition variations (subjects, speaker layouts, mixing variations, source locations, etc.) is demanding and requires significant domain knowledge. We developed the Binamix library to generate large amounts of binaural data for our work, but we believe it has many additional applications in spatial audio research. We decided to create our own direct binaural rendering pipeline instead of using an existing Ambisonics renderer. The reason for this is that we needed to be able to measure the sensitivity of our algorithms to a change in a single variable (azimuth, elevation, subject, speaker layout etc.) without having to account for any spatial distortions arising from the Ambisonics encode/decode process.

2.2 Library Overview

The tools include functions for binaurally mixing and rendering sources for any given azimuth and elevation, even if the SADIE II subject data does not have discrete impulse responses for the desired angles. In this instance, the tools will automatically generate suitable binaural impulse responses at render time by identifying the nearest available angles using Delaunay triangulation [12] and then using Euclidean distance measures to calculate the interpolation weights. Various interpolation modes can be selected, depending on user requirements, which will be described in more detail in section 6. This approach ensures that the rendered audio closely matches the intended spatial positioning.

The tools can also simulate binaurally rendered channel-encoded surround sound by simulating vector-based amplitude panning methods (VBAP) [13] within our binaural mixing pipeline. This simulates the rendering of sources at virtual speaker positions between the discrete speaker locations defined in a surround speaker layout. As such, Binamix allows for simulating channel-encoded surround mixes which have been binaurally rendered. This was motivated by the fact that much of the legacy content which is still in use today, exists in channel-encoded surround formats such as 5.1 and 7.1. These legacy formats will ultimately become the source format for direct binaural rendering when delivered over headphones. Because of this, it is necessary to be able to simulate a wide variety of these conditions when building codecs and quality metrics for binaural audio. The tools also support binaural rendering of real-world channel-encoded surround mixes, if you have access to them.

The library also includes a wide range of useful SADIE II wrapper functions which can be utilized to create custom pipelines and a range of example scripts to get started. Binamix also includes a loader script for the DSD100 Stems Database [14], which contains individual stems for vocals, drums, bass, and other instruments. However, the tools can be used with any audio content. This paper introduces the main concepts of Binamix and serves as a starting point for using the

library. Section 2.3 illustrates some usage examples of Binamix.

2.3 Applications

In general, the library allows you to achieve almost any task requiring programmatic and repeatable binaural mixing or rendering. Specific use cases include:

- Generating binaural and stereo mixes for general testing and evaluation.
- Simulating various binaural reference and test conditions for audio codec creation and evaluation.
- Simulating various binaural reference and test conditions for audio quality metric creation and evaluation.
- Generating large amounts of real-world binaural source renders or binaural mix renders for training machine learning models.
- Creating training datasets with a variety of conditions for:
  - Azimuth
  - Elevation
  - Subject ID (e.g., SADIE subjects; D1, D2, H3... H20)
  - IR type (HRIR or BRIR)
  - Speaker layout (e.g., none, 5.1, 7.1, 7.1.4, custom)
  - Mix Parameters (level, reverb, etc.)
  - Sample Rate (e.g., 44100, 48000, 96000)
  - Audio content (any audio you want to mix or render)

By offering access to repeatable, programmatic binaural mixing and rendering, the Binamix library supports the development of new technologies and applications in spatial audio.

3 Method and Design

Binamix is designed to emulate many of the basic functions in a commercial spatial audio processing pipeline. Figure 1 shows a simplified processing pipeline overview with Binamix components depicted in green. The components we do not currently provide are depicted in gray for reasons discussed in section 2.1.

3.1 Mixing Architecture

The mixing architecture, shown in Figure 2, allows you to render single sources or binaurally mix multiple sources. When mixing multiple sources, a simple `TrackObject` class allows you to specify track name, level, reverb, azimuth, and elevation. Level and reverb are specified in the normalized range 0 to 1. Azimuth can be specified either in the range of 0 to 360 degrees or as positive and negative angles. Positive angles are interpreted in the counter-clockwise direction and negative angles are interpreted in the clockwise direction. Elevation is best specified in the range -90 to +90 degrees but the library will convert values outside that range. If the SADIE II Database or selected speaker layout does not contain the discrete angle you specify, Binamix will automatically generate a suitable IR using the interpolation methods described in section 3.3. You can specify as many tracks as you like in this manner. The tracks are then collected into an array of tracks.

Figure 3 shows how the `TrackObject` is used and how it is passed to the binaural mixer function, `mix_tracks_binaural`. This function takes an array of track objects, the subject ID, sample rate, IR Type, speaker layout, interpolation mode, and reverb type as arguments. There is also a separate stereo mixer function which can be used in a very similar manner but uses constant power panning instead of azimuth and elevation. Consult the Binamix documentation for more information.

Referring to the function arguments depicted in Figure 3, subject IDs and IR types are discussed in section 2, and full details can be found in [15]. There are 4 reverb models to choose from: 1 = Theatre, 2 = Office, 3 = Small Room, and 4 = Meeting Room. These IRs were taken from the Achen Impulse Response Database [16]. The surround speaker layouts are discussed further in section 3.2.

3.2 Surround Rendering and Layout Simulation

The `render_surround_to_binaural` function works similarly to the `mix_tracks_binaural` function except that it uses an array of channel-encoded surround tracks as the input. Figure 4 shows how the function is used. The function takes the input and output layouts as arguments (e.g., input = 7.1.4, output = 5.1.4) and binaurally renders the audio at the correct

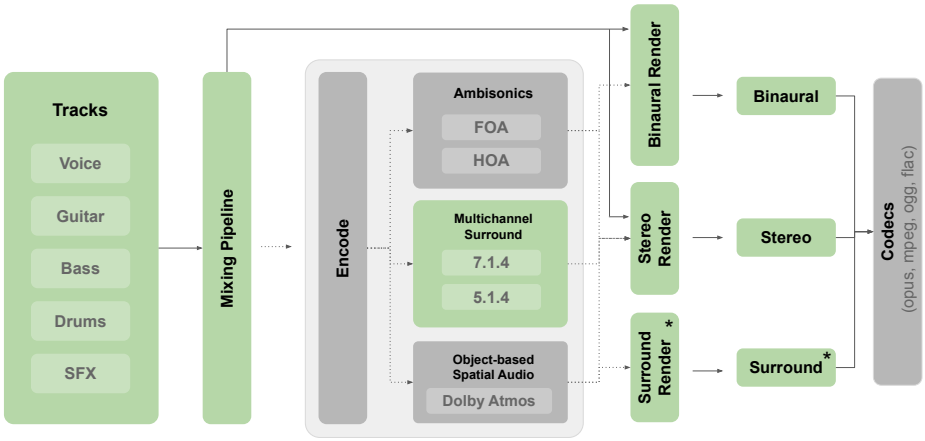


Fig. 1: Binamix system overview. Blocks in green are currently supported.

angles for the chosen layout. The angles for each layout are defined in the `surround_utilities.py` file which currently supports the following layouts: `['5.1', '5.1.4', '5.1.2', '7.1', '7.1.4', '7.1.2', '9.1.4', '9.1.2', '9.1']`. If the input and output layouts are the same, the renderer will use discrete angles for the specified surround layout. If the output format differs, the rendering function will use interpolation methods described in section 3.3 to approximate a down-mix. The channel order and naming conventions come from the SMPTE ST 2067-8 standard [17] and Dolby "Additional Channels For Immersive Audio" [18].

3.3 Interpolation Method

When using surround layout simulation with the binaural mixer, the tools will use IR interpolation to simulate the VBAP method [13] and render the source at the virtual speaker positions between the discrete angles available for a speaker layout. This allows you to create simulated surround mixes (binaurally rendered) for any given speaker layout described in 3.2. This should not be confused with the dedicated channel-encoded surround rendering function described in 3.2, which does not use interpolation unless down-mixing to a lower-order speaker layout. All of this logic is handled automatically, but it is important to understand the distinction. To clarify,

- `mix_tracks_binaural` allows you to create your own binaural mixes which can simulate the limitations of the channel-encoded surround (e.g., amplitude-only panning).
- `render_surround_to_binaural` allows you to binaurally render an existing real-world channel-encoded surround mix.

Interpolation may also be required even when speaker layout simulation is not being used. For example, there are only 50 BRIR angles available for the human subjects within SADIE II compared to the 8802 angles available for dummy subject HRIRs. So in the instance where a SADIE subject does not contain an angle within 2 degrees of the desired angle, a suitable interpolated angle is generated using proximal angles.

3.3.1 Delaunay Triangulation

Finding proximal angles to use for interpolation is slightly more challenging than finding the three nearest neighbors because the sphere is not uniformly sampled across azimuth and elevation. For human subject BRIRs in SADIE II, a 50-point Lebedev Quadrature [19] distribution is used, and for HRIRs, higher order sampling is used but azimuth resolution is always significantly higher than elevation resolution. As a result, a nearest neighbor search will almost always return 3 points on a line instead of the triangle within which the

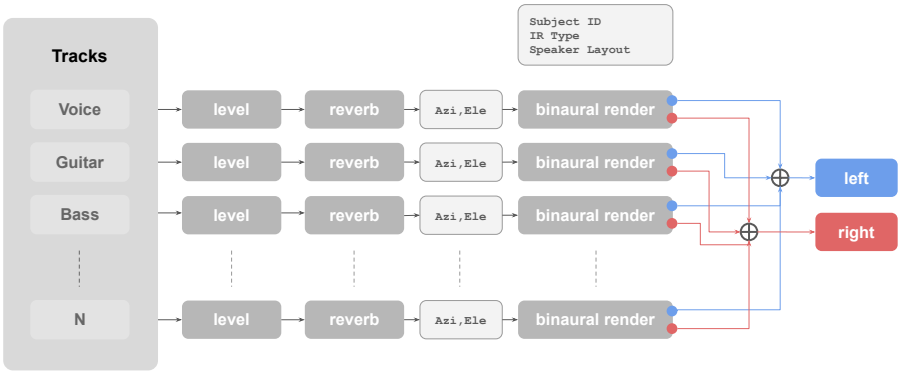


Fig. 2: Binamix signal flow.

```
# for binaural mixing
track1 = TrackObject('vocals', azimuth=45.0, elevation=0.0, level=0.5, reverb=0.2, audio=audio_data)
track2 = TrackObject('guitar', azimuth=-45.0, elevation=0.0, level=0.7, reverb=0.1, audio=audio_data)

tracks = [track1, track2]

output_binaural = mix_tracks_binaural(tracks, 'H3', 44000, 'HRIR', '7.1', mode="auto", reverb_type = "1")
```

Fig. 3: Example of binaural mixing

point lies. To overcome this problem, we used a modified Delaunay triangulation. The Delaunay method is normally used for 2D and 3D triangulation with cartesian coordinates but our modified version considers the angles and elevations to be an equirectangular projection of the sphere which is valid to find the bounding triangle for the desired angle. However, it is not a valid domain to calculate distances, so we convert the bounding angles into the cartesian domain. Then we calculate the Euclidean distance from the desired angle to the bounding angles in order to calculate interpolation weights for generating the required IR. The spherical domain is periodic but an equirectangular projection is not, so a desired angle can fall outside the Delaunay mesh. If this happens, our method will rotate the projection by 180 degrees, vertically and horizontally if necessary, and will always return a bounding triangle. Figure 5 shows the Delaunay triangulation of a desired angle within a 50-point Lebedev distribution and a 7.1.4 distribution. The `delaunay_triangulation` function in the Binamix library generates these plots by setting `plots=True` in the arguments.

3.3.2 Interpolation Modes

The library provides 5 interpolation modes, each of which has a slightly different effect on the resulting render. Figure 6 depicts which points are used for each of the modes which are described as follows:

- **nearest**: Uses the nearest available discrete angle, thus avoiding the use of interpolation at all. This mode guarantees the use of a real IR but compromises spatial fidelity for low-density point distributions.
- **planar**: Creates a weighted interpolation between the nearest neighbors on the closest elevation plane. This compromises elevation accuracy but suffers less spectral coloration than three-point interpolation.
- **two_point**: Creates a weighted interpolation between the two points on the nearest line to the desired angle. The function automatically chooses between azimuth or elevation interpolation based on the point distribution or speaker layout.

```
# for channel encoded surround rendering
output = render_surround_to_binaural(surround_container, 48000, 'H3', 'HRIR', '7.1.4', '5.1', mode="auto")
```

Fig. 4: Example of channel-encoded surround rendering

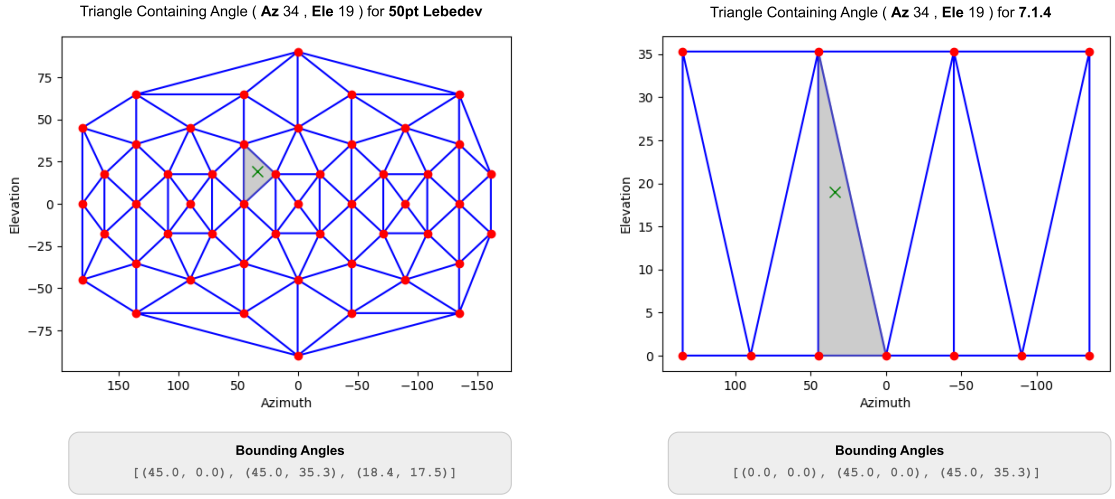


Fig. 5: Delaunay Triangulation on two spatial distributions: 50-point Lebedev (left) and 7.1.4 layout (right).

- **three_point**: Uses three-point weighted interpolation between the bounding angles found using the Delaunay triangulation method described above.
- **auto**: Automatically selects the best interpolation method based on which one achieves the closest Euclidean approximation to the desired angle.

In general, using the fewest points possible for interpolation leads to the most favorable outcome since each point contributes its own ITD, ILD, and spectral cues to the interpolated IR. Furthermore, interpolation fidelity is inherently limited by the available point density.

3.4 Example Scripts

The library also includes several example scripts which briefly demonstrate how to use the library for various tasks. We hope that they serve as useful templates for your own projects. Example scripts include:

- Basic example of binaural mixing with 4 tracks.
  `binaural_mixer_example.py`
- Example of rendering a multichannel surround mix to binaural.
  `render_surround_encoded_wav_example.py`
- Basic example of stereo mixing with 4 tracks.
  `stereo_mixer_example.py`
- Render a single mix for all available speaker layouts.
  `mix_for_all_speaker_layouts_example.py`
- Transcode audio files to Opus format.
  `opus_transcodes_example.py`
- Example of Delaunay triangulation with plot.
  `triangulation_with_plot_example.py`
- Example of generating a binaural dataset.
  `generate_binaural_dataset_example.py`

4 Conclusion

Binamix provides a modular and extensible framework for generating high-quality binaural audio datasets at

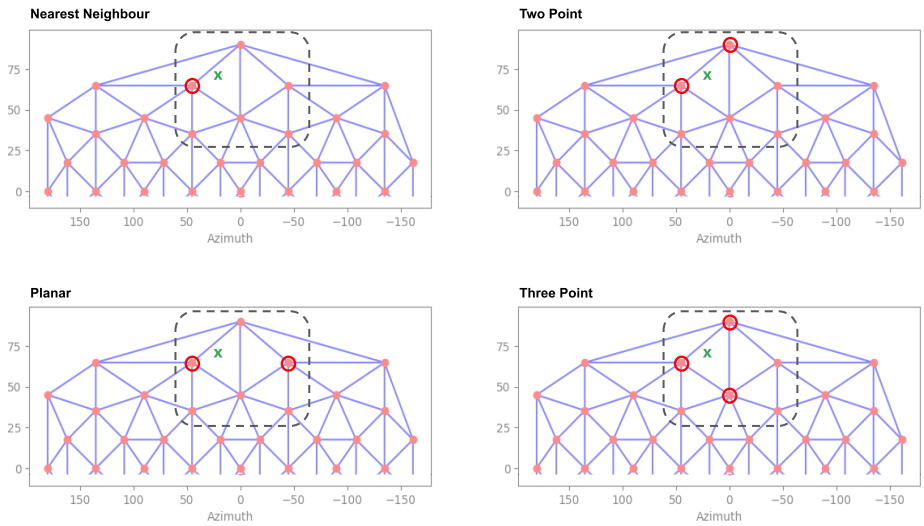


Fig. 6: Interpolation Modes: Nearest Neighbor, Two Point, Three Point and Planar

scale. By integrating robust binaural rendering, flexible mixing pipelines, and support for multiple speaker layouts, the library simplifies the process of simulating realistic binaural audio conditions. The inclusion of interpolation methods and layout-aware rendering ensures accuracy even when discrete IRs are unavailable, and the modular design allows for easy customization and integration into larger experimental workflows.

The library is not designed to be a fully functional spatial audio production suite, but there are still some features we plan to add in the future. Currently, the library only supports mono reverb as a process prior to HRIR/BRIR convolution. This is suitable for our current needs but a multichannel reverb would allow for a more flexible simulation of real-world production conditions. The library currently only supports static source positions but we plan to add support for dynamic source movement in the future.

We hope this tool accelerates research in spatial audio coding, quality assessment, and machine learning by lowering the barrier to reproducible and customizable binaural dataset creation.

5 Acknowledgments

This work was conducted with a research grant from Taighde Éireann – Research Ireland co-funded under the European Regional Development Fund under Grant Numbers 12/RC/2289_P2 and a gift from Google. For the purpose of Open Access, the author has applied a CC BY public copyright license to any Author Accepted Manuscript version arising from this submission.

References

[1] Dewey, C., Moore, A., and Lee, H., "Practitioners' Perspectives on Spatial Audio: Insights into Dolby Atmos and Binaural Mixes in Popular Music," *AES: Journal of the Audio Engineering Society*, 72(7/8), pp. 504–516, 2024, ISSN 0004-7554.

[2] Alliance for Open Media, "Immersive Audio Metadata Format (IAMF)," `https://aomediacodec.github.io/iamf/`, 2025, accessed: 2025-01-28.

[3] Rane, M., Coleman, P., Mason, R., and Bech, S., "Survey of User Perspectives on Headphone Technology," *Journal of the Audio Engineering Society*, (10556), 2022.

[4] Narbutt, M., Skoglund, J., Allen, A., Chinen, M., Barry, D., and Hines, A., "AMBIQUAL: Towards a Quality Metric for Headphone Rendered Compressed Ambisonic Spatial Audio," *Applied Sciences*, 10(9), p. 3188, 2020, doi: 10.3390/app10093188.

[5] Perez, A. and Contributors, "pysofaconventions: A Python Library for Handling SOFA Files," https://github.com/andresperezEUT/pysofaconventions, 2025, accessed: 2025-01-28.

[6] Franck, A., Costantini, G., Pike, C., and Fazi, F. M., "An Open Realtime Binaural Synthesis Toolkit for Audio Research," in *Proceedings of the Audio Engineering Society 144th Convention*, Milano, Italy, 2018, engineering Brief.

[7] Barry, D., Panah, D. S., Ragano, A., Skoglund, J., and Hines, A., "Binamix: A Python Library for Generating Binaural Audio Datasets," https://github.com/QxLabIreland/Binamix/, 2025, accessed: 2025-04-01.

[8] Hines, A., Skoglund, J., Kokaram, A. C., and Harte, N., "ViSQOL: An Objective Speech Quality Model," *EURASIP Journal on Audio, Speech, and Music Processing*, 2015(1), pp. 1–18, 2015, doi:10.1186/s13636-015-0054-9.

[9] Ragano, A., Skoglund, J., and Hines, A., "NOMAD: Unsupervised Learning of Perceptual Embeddings For Speech Enhancement and Non-Matching Reference Audio Quality Assessment," in *ICASSP 2024 - 2024 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 1011–1015, 2024, doi:10.1109/ICASSP48485.2024.10448028.

[10] Ragano, A., Skoglund, J., and Hines, A., "SCOREQ: Speech Quality Assessment with Contrastive Regression," in *Advances in Neural Information Processing Systems*, volume 37, pp. 105702–105729, 2024.

[11] Harte, N., Gillen, E., and Hines, A., "TCD-VoIP, a research database of degraded speech for assessing quality in VoIP applications," *2015 7th International Workshop on Quality of Multimedia Experience, QoMEX 2015*, 2015, doi:10.1109/QoMEX.2015.7148100.

[12] Elshakhs, Y. S., Deliparaschos, K. M., Charalambous, T., Oliva, G., and Zolotas, A., "A Comprehensive Survey on Delaunay Triangulation: Applications, Algorithms, and Implementations Over CPUs, GPUs, and FPGAs," *IEEE Access*, 12, pp. 12562–12585, 2024, doi:10.1109/ACCESS.2024.3354709.

[13] Pulkki, V., "Virtual Sound Source Positioning Using Vector Base Amplitude Panning," *Journal of Audio Engineering Society*, 45(6), pp. 456–466, 1997.

[14] Liutkus, A., Stöter, F.-R., Rafii, Z., Kitamura, D., Rivet, B., Ito, N., Ono, N., and Fontecave, J., "The 2016 Signal Separation Evaluation Campaign," in P. Tichavský, M. Babaie-Zadeh, O. J. Michel, and N. Thirion-Moreau, editors, *Latent Variable Analysis and Signal Separation - 12th International Conference, LVA/ICA 2015, Liberec, Czech Republic, August 25-28, 2015, Proceedings*, pp. 323–332, Springer International Publishing, Cham, 2017.

[15] Armstrong, C., Thresh, L., Murphy, D., and Kearney, G., "A Perceptual Evaluation of Individual and Non-Individual HRTFs: A Case Study of the SADIE II Database," *Applied Sciences*, 8(11), p. 2029, 2018, doi:10.3390/app8112029.

[16] Institute of Communication Systems, RWTH Aachen University, "Aachen Impulse Response (AIR) Database," https://www.iks.rwth-aachen.de/en/research/tools-downloads/databases/aachen-impulse-response-database/, 2009, accessed: 2025-03-27.

[17] "SMPTE ST 2067-8: Interoperable Master Format — Common Audio Labels," 2013.

[18] Dolby Laboratories, "Additional Audio Channels and Soundfields for Immersive Audio," Technical report, Dolby Laboratories, 2018, accessed: 2025-03-28.

[19] Lecomte, P., Gauthier, P.-A., Langrenne, C., Berry, A., and Garcia, A., "A Fifty-Node Lebedev Grid And Its Applications To Ambisonics," *Journal of the Audio Engineering Society*, 64, pp. 868–881, 2016, doi:10.17743/jaes.2016.0036.