AMBIVIZ: A PYTHON TOOLBOX FOR AMBISONIC AUDIO AND 360-DEGREE VIDEO VISUALIZATION

Jinyue GUO (郭锦岳) (jinyue.guo@imv.uio.no) (0009-0003-7438-1483) [1,2],
Maham RIAZ (maham.riaz@imv.uio.no) (0000-0002-4636-7436) [1,2], and
Alexander R. JENSENIUS (a.r.jensenius@imv.uio.no) (0000-0001-6171-8743) [1,2]

[1] *RITMO Centre for Interdisciplinary Studies in Rhythm, Time and Motion*, **University of Oslo**, Oslo, Norway
[2] *Department of Musicology*, **University of Oslo**, Oslo, Norway

ABSTRACT

This paper introduces the open-source Python toolbox *Ambiviz*, which focuses on visualizing Ambisonic audio and 360-degree video recordings. While many audio plug-ins and proprietary programs support different forms of spatial audio visualization, we found two main issues when using them for research purposes: (1) existing plug-ins and programs do not support a programming interface, making it difficult to either batch process datasets or modify the visualizations according to different needs; (2) current visualization graphs cannot represent stationary indoor environments well. *Ambiviz* supports batch processing, easily modifiable via the matplotlib backend, and includes custom visualizations for indoor environments: the audio energy map, anglegrams, and directograms. These visualizations can show different spatial-temporal aspects of immersive audio–video recordings. The source code for *Ambiviz* is available at `https://github.com/fisheggg/ambiviz`.

1. INTRODUCTION

Spatial audio, and more specifically Ambisonic audio [1, 2], has attracted attention in recent years from a broader audience beyond sound engineers and music technologists. Artists, architects, and machine learning researchers have begun using Ambisonic audio for various applications, such as immersive artworks [3–5], urban planning [6], and multimodal machine learning models [7–12].

While there are plenty of audio engineering tools for Ambisonics processing and visualization, many rely on being hosted in commercial software, such as digital audio workstations (DAWs) or music programming environments. DAW plug-ins for Ambisonics visualization, such as the *EnergyVisualizer* in the IEM plug-in suite [1] and the *ab Advanced Visualiser HOA* [2], are accurate and professional in Ambisonics visualization with many options in decoding algorithms and other features. However, they are not designed to quickly get an overview of a large number of Ambisonics recordings, which is a common need for researchers and artists performing many or long field recording sessions. Furthermore, the lack of simple programming APIs for existing plug-ins makes it difficult to control parameters, output the generated visualizations to files, and batch-process datasets.

Conversely, existing Python toolboxes such as *librosa* [13] and *torchaudio* [14, 15] provide rich functionality for temporal and spectral analysis but do not support Ambisonic audio or spatial analysis. *librosa* is ideal for quickly generating visualizations such as spectrograms, mel-spectrograms, and tempograms. *torchaudio* is ideal for generating large batches of audio features with GPU acceleration, which is well-suited for producing input features for neural network training on large datasets. Moreover, the simple Python interface and the open-source nature of these tools have significantly reduced the requirements for audio processing. However, they do not currently support Ambisonic input, although they do support some form of multi-channel audio. The Python libraries also lack support for spatial visualization methods designed for spatial audio, such as the *audio energy map* (AEM) [16].

Finally, even though both Ambisonic audio and 360-degree video capture related spatial-temporal information, we did not find any existing tool that can systematically create combined workflows for the two media types (and other related modalities). Drawing on our experience using Ambisonic audio and 360-degree video to record and analyze indoor soundscapes [17–19], we propose *Ambiviz* as a general-purpose Python toolbox. In addition to a set of backend tools for importing and processing files, the toolbox includes functions for creating audio energy maps overlaid with flattened 360-degree images, anglegrams for visualizing a moving, pointed sound source, and directograms for visualizing ambient sounds in indoor environments.

2. FUNCTIONALITIES OF AMBIVIZ

2.1 Design principles

The *Ambiviz* toolbox is developed as an all-in-one library for spatial visualization of Ambisonic audio and for combining it with 360-degree video. The design aim has been

[1] `https://plugins.iem.at/#tab-EnergyVisualizer`
[2] `https://www.audiobrewers.com/plugins/p/ab-advanced-visualiser-hoa`

Copyright: © 2026. This is an open-access article distributed under the terms of the Creative Commons Attribution 3.0 Unported License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original author and source are credited.



Figure 1. A frame of an audio energy map video generated with `ambiviz.aem()`. The image is an equirectangular display of a 360-degree video recording. The redness indicates the location of the highest audio energy from the corresponding Ambisonic audio. In this frame, the sound source is near the white drawer.

to make the toolbox accessible to students, researchers, and artists without requiring advanced programming skills. Therefore, the toolbox is designed to make the API as simple as possible. The minimum requirement to generate a visualization is only to provide a path to the audio file, with all audio processing and plotting parameters predefined.

For more advanced users, *Ambiviz* also support GPU-accelerated Ambisonic audio processing and adjusting of the visualization parameters. Since Ambisonics recordings are often used to capture soundscapes and concerts, users may need to process hours-long, high-fidelity, multi-channel audio files. Therefore, the functions are designed to accept a simple boolean parameter `gpu` that enables all computations to be performed with PyTorch GPU tensors.

We also see the need to support machine learning-based workflows, in which some of the calculated features, such as the audio energy map, are often used as input features of neural networks [9, 11, 12]. For these purposes, in addition to GPU acceleration, an efficient, silent API is important for large-scale batch processing. Therefore, a boolean parameter `verbose` is provided to toggle a verbose mode with progress bars or to let the function be completely silent.

Finally, we have aimed to provide readable code and cross-platform compatibility. The source code is openly available and developed with modern Python practices, using *uv* for dependency management, *ruff* for formatting, and GitHub Actions for the CI workflow. The toolbox is compatible with Python versions higher than 3.9.

As for now, *Ambiviz* includes three visualization and analysis functionalities: the `ambiviz.anglegram()` and `ambiviz.directogram()` for spatial analysis of audio, and the `ambiviz.aem()` that combines Ambisonic audio with 360-degree video.

2.2 Backend

The *Ambiviz* toolbox relies on the following libraries:

- PyTorch [20] for tensor processing and GPU acceleration
- Torchaudio [14, 15] and FFmpeg [3] for audio processing
- MGT-Python [21, 22], OpenCV-Python [23] and FFmpeg[3] for 360-degree video processing
- Matplotlib [24] and SciencePlots [25] for plotting and styling
- The Python implementation of Ambisonics decoding algorithm by Morgado et al. [16]

The functions in *Ambiviz* will first load input audio and video into `torch.Tensor` with *torchaudio* and *MGT-Python* with optional trimmings and stitchings. The core Ambisonics decoding algorithm is modified from Morgado et al. [16] with all matrices been changed to `torch.Tensor` as well to support GPU acceleration. The algorithms for each *Ambiviz* functionality is performed in native Python. Finally, the stable figure visualizations are created with *Matplotlib* with an academic style by *SciencePlots* [25], and the video visualizations are created with *OpenCV-Python* [26].

[3] `https://ffmpeg.org/`

2.3 Audio energy map

The *audio energy map* (AEM), as shown in Figure 1, visualizes the spatial distribution of audio. It is often combined with 360-degree video to show an immersive audio–video-scape. Formalized by Morgado et al. [16], AEM is based on using a virtual speaker matrix to calculate the audio energy at each point on the sphere and map it to the 360-degree video. The audio energy map can be used for both qualitative analysis of indoor environments [17] and as inputs for neural networks [12].

The audio energy map (AEM) is calculated in the following steps:

1. Select a set of directions $\boldsymbol{\Theta} = \{\boldsymbol{\theta}\} = \{(\varphi, \vartheta)\}$, where $\varphi \in \{-180°, -178°, \cdots, 176°, 178°\}$ denotes the azimuth (horizontal) angle, and $\vartheta \in \{-90°, -88°, \cdots, 88°, 90°\}$ denotes the elevation (vertical) angle (step size is adjustable);

2. For each $\boldsymbol{\theta}$, decode the waveform $x[\boldsymbol{\theta}, n] = \sum_{h=0}^{H=1} \sum_{d=-h}^{h} Y_h^d(\varphi, \vartheta)\phi_h^d[n]$, where h is the Ambisonics order, d is the Ambisonics degree, Y_h^d is the spherical harmonic component, and $\phi_h^d[n]$ is the corresponding coefficient;

3. The AEM is defined as the sliding RMS energy of each direction: $AEM[\boldsymbol{\theta}, m] = \sqrt{\frac{1}{N}\sum_{n=m\times N}^{m\times N+N-1}(x[\boldsymbol{\theta}, n])^2}$, where m is the frame index and N is the frame length.

The concept of AEM has evolved since the invention of Ambisonics technology, which was introduced to machine learning by Morgado et al. [16], and has been widely adopted [9, 11, 12, 27]. However, the code for generating AEMs differs across these studies, with no simple, formalized way to generate AEMs in Python. In *Ambiviz*, we formalized the code for generating AEM from [16] into a simple function-based API. The code is also modified to support GPU acceleration via PyTorch. Lastly, since Ambisonics and 360-degree video are often recorded by two separate devices, we add the `audio_offset`, `video_offset`, and `duration` parameters to allow the user to sync the two media files and trim the output. An AEM can be generated using a single function call:

```
import ambiviz
ambiviz.aem(
  audio_path = "path/to/audio.wav",
  video_paths = ["multiple/videos.mp4",
                 "are/merged.mp4"],
  output_path = "path/to/output_video",
  duration = 30.0, # optional
  audio_offset = 1.0, # optional
  video_offset = 15.2, # optional
)
```

The generated video will also include the omnidirectional channel from the input Ambisonics file.

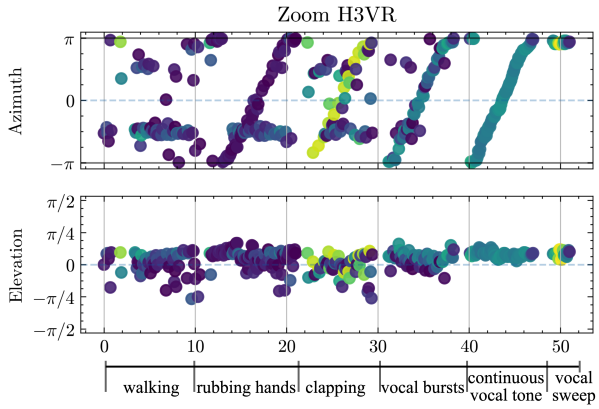


Figure 2. An anglegram generated from an Ambisonic audio recording by `ambiviz.anglegram()`. The visualization illustrates the sound captured of a person moving around the microphone in circles, making different sounds for each circle. The top display (azimuth) shows how the visualization effectively displays the horizontal displacement. The bottom display (elevation) correctly visualizes that the sound source was kept horizontally stable throughout the recording.

2.4 Anglegram

The *anglegram* is designed to demonstrate the temporal evolution of spatial energy. As shown in Figure 2, the horizontal axis shows the time, while the vertical axis shows the direction of arrival of the highest audio energy. The anglegram is a useful tool for generating an overview of the motion of a single-pointed sound source. The anglegram is calculated in the following steps:

1. Calculate $AEM[\boldsymbol{\theta}, m] = AEM[\varphi, \vartheta, m]$ according to the steps described in Sec 2.3.

2. For each m, find the argmax $\boldsymbol{\theta}$. The direction with the highest energy is labeled as the dominant sound source for that instant: $A[m] = (\varphi_m, \vartheta_m, E_m) = (\arg\max_{\varphi,\vartheta} AEM[\varphi, \vartheta, m], \max_{\varphi,\vartheta} AEM[\varphi, \vartheta, m])$. We define the resulting sequence $A[m]$ as the anglegram.

In code implementation, the anlegram is calculated from the AEM of input audio. Additional parameters in `ambiviz.anglegram()` allow changing the audio frame length, hop length, azimuth and elevation resolution, and the dB range to be plotted in the figure. The call for generating an anglegram can be:

```
import ambiviz
ambiviz.anglegram(
  audio_path = "path/to/audio.wav",
  save_path = "path/to/output.png",
  fps = 20 # optional,
  aem_width = 90, # optional
  aem_height = 45, # optionals
)
```

2.5 Directogram

The *directogram* is designed to visualize the spatial characteristics of auditory stimuli in a room. As shown in Figure 3, the directogram summarizes the spatial energy at the maximum direction of each audio frame, and then combines the adjacent energies into bins for every 5 degrees. As we can see, different indoor environments have very different spatial distribution of audio energy, which can describe the characteristics of the room in addition to traditional spectral and temporal visualizations. Some rooms (SS#001) have spread directions of the loudest sound sources, whereas other rooms (SS#051, SS#126) have pointed sound sources from very specific angles.

The directogram $E_{amb}[\phi]$ is calculated in the following steps:

1. Calculate $AEM[\boldsymbol{\theta}, m] = AEM[\varphi, \vartheta, m]$, where φ is the azimuth angle, ϑ the elevation angle, and m the frame index, according to the steps described in Sec 2.3.

2. Calculate the audio energy at the horizontal plane by taking the AEM at $\vartheta = 0°$: $AEM_h[\varphi, m] = AEM[\varphi, \vartheta = 0°, m]$.

3. Calculate the directogram $D[\varphi]$ by summing the maximum directional energy of each time frame: $D[\varphi] = \sum_{m \in M} AEM_h[\varphi | \varphi = \text{argmax}(AEM_h[\varphi, m]), m]$.

The code implementation is derived from the audio energy map. Additional parameters include audio frame length, hop length, azimuth, and elevation resolution. The call to generate a directogram can be:

```
import ambiviz
ambiviz.directogram(
  audio_path = "path/to/audio.wav",
  save_path = "path/to/output.png",
  gpu = True, # optional
)
```

3. USE CASES

The three visualization functions can reveal different spatial characteristics of Ambisonic audio recordings. The figures can be used directly to compare recordings or as input features to machine learning models. Moreover, they can be combined with other data types—such as flattened 360-degree images in the audio energy map—to perform cross-modal analyses. Here, we demonstrate two additional use cases for *Ambiviz*.

3.1 Using directograms to summarize spatial audio energy of indoor environments

Directograms can demonstrate the spatial characteristics of indoor environments. Figure 3 shows how directograms can compare the spatial differences of different indoor envrionments. Additionally, we can investigate the effect of

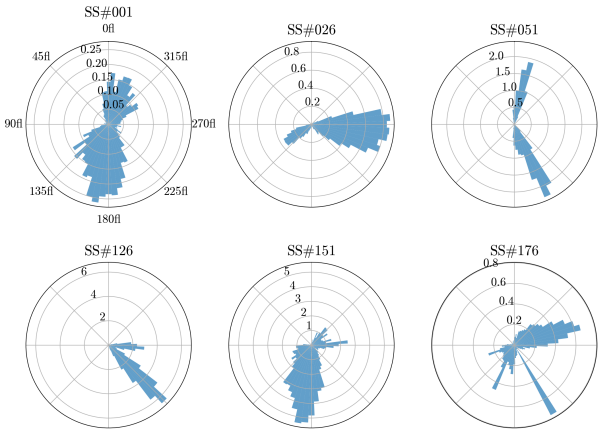


Figure 3. Directograms generated by `ambiviz.directogram()`. Each subplot summarises a 10-minute recording of ambient sound in a room. 0° is the front in the horizontal plane, and 90°, 180°, 270° are the left, back, right, respectively.

such characteristics on human beings by combining accelerometer data of perceivers while standing still in the room [19]. Figure 4 shows a combined plot with a directogram and motion capture data of the swaying patterns of a person.

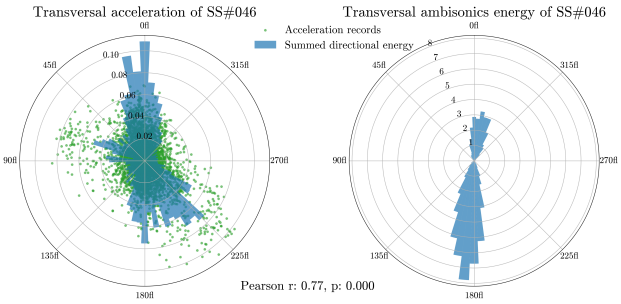


Figure 4. Directionality of micromotion swaying patterns (left) and directogram (right). Such plots can be used to inspect similarities in motion patterns and audio patterns.

3.2 Using anglegram to evaluate spatial accuracy of capturing devices

The anglegram can demonstrate the motion of a pointed sound source over time and is therefore useful for evaluating the spatial resolution of different Ambisonic microphones or other devices, such as 360 cameras that claim to capture some sort of "spatial audio" [22], as shown in Figure 2. For a more precise quantitative analysis, the anglegram can be used with a motion capture system: the Ambisonics capturing device and the sound source are both marked with motion capture markers, which can then be used as the ground truth direction of arrival (DoA) [18], as shown in Figure 5.

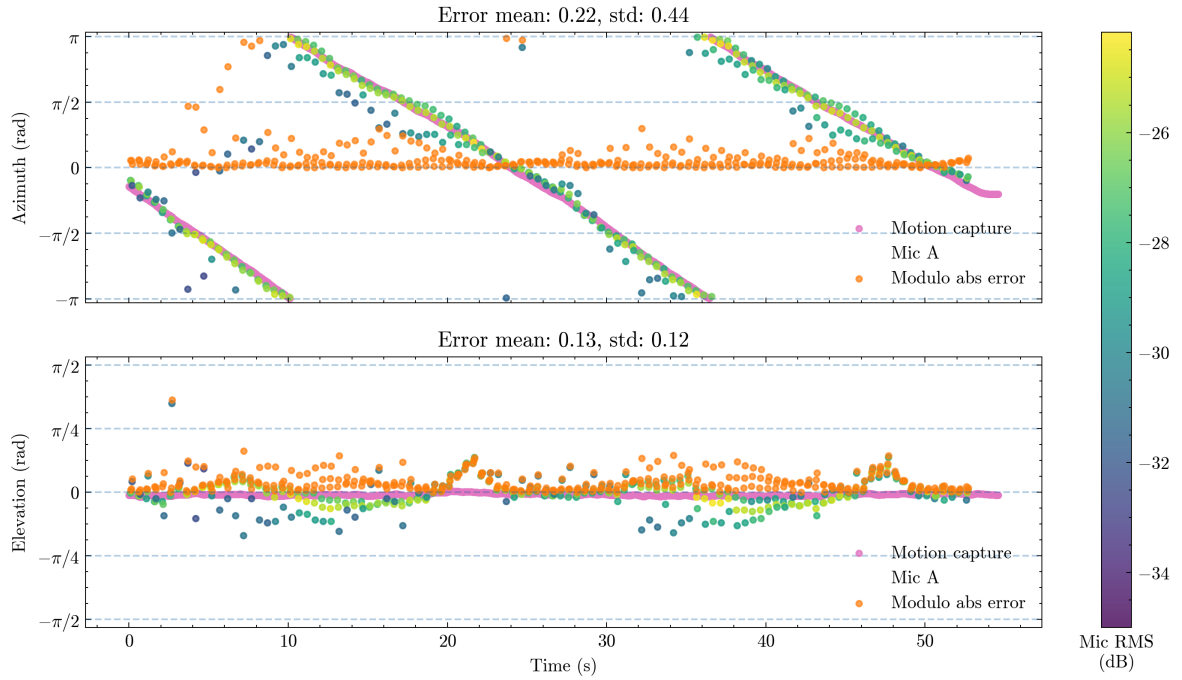


Figure 5. Using anglegram and motion capture to evaluate the spatial accuracy of Zoom H3-VR [18]. Pink dots are motion capture data used as ground truth DoA, purple to yellow dots are anglegram from the microphone, and orange dots are the error.

4. CONCLUSIONS AND FUTURE WORK

We developed *Ambiviz*, an open-source Python toolbox for Ambisonics and 360° video visualization. It is designed for students, artists, and researchers without requiring advanced knowledge of programming or signal processing. The toolbox currently contains three main visualization methods, `aem()`, `anglegram()`, and `directogram()`, each demonstrating different spatial characteristics of the Ambisonic audio recording. These functions can be used as qualitative visualizations of the recordings, as input features (with GPU acceleration) for machine learning models, or combined with other data for a cross-modal analysis.

Future work includes enhancing existing functions and adding more visualization methods. For existing functions, enhancing the performance and robustness is important. We are working on supporting spatial audio from more devices, including the "almost-Ambisonic" audio streams from GoPro MAX 360 cameras, first-order Ambisonics from the Soundfield SPS200 microphone, and higher-order Ambisonics (HOA) from the ZYLIA ZM-1 HOA microphone. Currently, only the `aem()` function supports HOA. Some additional parameters, such as rotation of the Ambisonics sphere, would be useful for some recordings. For additional visualization methods, we are investigating spectral visualizations and better combinations with other data types, including physiological data.

Acknowledgments

This project is supported by the Research Council of Norway through projects 262762 (RITMO), 324003 (AMBIENT), and 322364 (fourMs). We also thank Jingxuan Yue for assistance in testing and developing.

5. REFERENCES

[1] F. Zotter and M. Frank, *Ambisonics: A Practical 3D Audio Theory for Recording, Studio Production, Sound Reinforcement, and Virtual Reality*, ser. Springer Topics in Signal Processing. Cham: Springer International Publishing, 2019, vol. 19.

[2] J. Ahrens, "Introduction to Ambisonics, Part 1: The Part With No Math," Dec. 2025, arXiv:2512.07570 [eess].

[3] N. Barrett, "The perception, evaluation and creative application of high order ambisonics in contemporary music practice," *Ircam Musical Research Residency report*, 2012.

[4] ——, "Spatial music composition," in *3D Audio*. Routledge, 2021, pp. 175–191.

[5] E. Bates, "The composition and performance of spatial music," Ph.D. dissertation, Trinity College Dublin, 2009.

[6] W. Kropp, J. Forssén, and L. Estévez-Mauriz, *Urban sound planning: The SONORUS project*. Chalmers University of Technology, Division of Applied Acoustics, 2016.

[7] H. Liu, T. Luo, K. Luo, Q. Jiang, P. Sun, J. Wang, R. Huang, Q. Chen, W. Wang, X. Li, S. Zhang, Z. Yan, Z. Zhao, and W. Xue, "OmniAudio: Generating

Spatial Audio from 360-Degree Video," Jun. 2025, arXiv:2504.14906 [eess].

[8] X. Jiang, H. Gamper, and S. Braun, "Sci-Phi: A Large Language Model Spatial Audio Descriptor," *IEEE Open Journal of Signal Processing*, vol. 7, pp. 276–284, 2026.

[9] J. Kim, H. Yun, and G. Kim, "Towards Scene-Aware Video-to-Spatial Audio Generation," *International Journal of Computer Vision*, vol. 134, no. 4, p. 184, Mar. 2026.

[10] G. Yuksel, M. v. Gerven, and K. v. d. Heijden, "GRAM: Spatial general-purpose audio representation models for real-world applications," Nov. 2025, arXiv:2506.00934 [cs].

[11] F.-Y. Chao, C. Ozcinar, L. Zhang, W. Hamidouche, O. Deforges, and A. Smolic, "Towards Audio-Visual Saliency Prediction for Omnidirectional Video with Spatial Audio," in *2020 IEEE International Conference on Visual Communications and Image Processing (VCIP)*, Dec. 2020, pp. 355–358.

[12] Q. Yang, Y. Li, C. Li, H. Wang, S. Yan, L. Wei, W. Dai, J. Zou, H. Xiong, and P. Frossard, "SVGC-AVA: 360-Degree Video Saliency Prediction with Spherical Vector-Based Graph Convolution and Audio-Visual Attention," *IEEE Transactions on Multimedia*, pp. 1–16, 2023, conference Name: IEEE Transactions on Multimedia.

[13] B. McFee, C. Raffel, D. Liang, D. Ellis, M. McVicar, E. Battenberg, and O. Nieto, "librosa: Audio and Music Signal Analysis in Python," in *Python in Science Conference*, Austin, Texas, 2015, pp. 18–24.

[14] Y.-Y. Yang, M. Hira, Z. Ni, A. Astafurov, C. Chen, C. Puhrsch, D. Pollack, D. Genzel, D. Greenberg, and E. Z. Yang, "Torchaudio: Building blocks for audio and speech processing," in *ICASSP 2022-2022 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2022, pp. 6982–6986.

[15] J. Hwang, M. Hira, C. Chen, X. Zhang, Z. Ni, G. Sun, P. Ma, R. Huang, V. Pratap, and Y. Zhang, "TorchAudio 2.1: Advancing speech recognition, self-supervised learning, and audio processing components for PyTorch," in *2023 IEEE Automatic Speech Recognition and Understanding Workshop (ASRU)*. IEEE, 2023, pp. 1–9.

[16] P. Morgado, N. Vasconcelos, T. Langlois, and O. Wang, "Self-Supervised Generation of Spatial Audio for 360 Video," Sep. 2018, arXiv:1809.02587 [cs, eess].

[17] M. Riaz, J. Guo, B. Serdar, and A. R. Jensenius, "Where is That Bird? The Impact of Artificial Birdsong in Public Indoor Environments," in *AM '25: Proceedings of the 20th International Audio Mostly Conference*. Coimbra, Portugal: Association for Computing Machinery (ACM), 2025.

[18] M. Riaz, J. Guo, C. Erdem, and A. R. Jensenius, "Where to Put That Microphone? A Study of Sound Localization in Ambisonics Recordings," in *Proceedings of the 17th International Symposium on Computer Music Multidisciplinary Research*, London, United Kingdom, Nov. 2025, conference Name: 17th International Symposium on Computer Music Multidisciplinary Research (CMMR 2025) ISBN: 9791097498061 Pages: 455-466 Publisher: Zenodo.

[19] J. Guo, J. Tørresen, and A. R. Jensenius, "Cross-modal Analysis of Spatial-Temporal Auditory Stimuli and Human Micromotion when Standing Still in Indoor Environments," in *Proceedings of the 17th International Symposium on Computer Music Multidisciplinary Research*, London, United Kingdom, Nov. 2025, conference Name: 17th International Symposium on Computer Music Multidisciplinary Research (CMMR 2025) ISBN: 9791097498061 Pages: 871-882 Publisher: Zenodo.

[20] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, A. Desmaison, A. Kopf, E. Yang, Z. DeVito, M. Raison, A. Tejani, S. Chilamkurthy, B. Steiner, L. Fang, J. Bai, and S. Chintala, "PyTorch: An Imperative Style, High-Performance Deep Learning Library," in *Advances in Neural Information Processing Systems*, vol. 32. Curran Associates, Inc., 2019.

[21] B. Laczkó and A. R. Jensenius, "Reflections on the Development of the Musical Gestures Toolbox for Python," in *Sound and Music Computing (SMC)*, 2021.

[22] J. Guo, M. Riaz, and A. R. Jensenius, "Comparing Four 360-Degree Cameras for Spatial Video Recording and Analysis," in *Sound and Music Computing (SMC)*, 2024.

[23] G. Bradski, "The opencv library." *Dr. Dobb's Journal: Software Tools for the Professional Programmer*, vol. 25, no. 11, pp. 120–123, 2000.

[24] J. D. Hunter, "Matplotlib: A 2D Graphics Environment," *Computing in Science & Engineering*, vol. 9, no. 3, pp. 90–95, May 2007.

[25] J. Garrett, E. Luis, H.-H. Peng, T. Cera, J. Borrow, M. Keçeci, S. Iyer, Y. Liu, and M. Gasanov, "garrettj403/SciencePlots: 2.1. 1," *Zenodo*, 2021.

[26] G. Bradski, "The OpenCV Library," *Dr. Dobb's Journal of Software Tools*, 2000.

[27] D. Zhu, K. Zhang, N. Zhang, Q. Zhou, X. Min, G. Zhai, and X. Yang, "Unified Audio-Visual Saliency Model for Omnidirectional Videos With Spatial Audio," *IEEE Transactions on Multimedia*, vol. 26, pp. 764–775, 2024, conference Name: IEEE Transactions on Multimedia.