Autogenous Spatialization for Arbitrary Loudspeaker Setups

Zeyu Yang
Audio Communication Group
TU Berlin
Berlin, Germany
zeyu.yang.1@campus.tu-berlin.de

Henrik von Coler
Audio Communication Group
TU Berlin
Berlin, Germany
voncoler@tu-berlin.de

***Abstract*—Autogenous spatialization is an experimental approach for distributing audio signals to loudspeaker systems, based on the signals' properties and the configuration of the loudspeaker setup. Time-domain and frequency-domain features of incoming audio streams are extracted and serve as control signals. The audio streams are then distributed to individual loudspeakers, based on a mapping between these control signals and properties of the loudspeakers, such as position and orientation. Results of the approach can range from a localizable panning between loudspeakers to alienating modulation effects and spatial sound synthesis applications. The proposed system is implemented as a C++ toolbox to allow the integration into various ecosystems for music production and sound design. External objects are provided for Pure Data, which allows the flexible testing and exploration of the spatialization approach.**

***Index Terms*—Autogenous Spatialization, Experimental Spatialization, Spatial Sound Synthesis, Pure Data**

I. Introduction

In the context of music production and performance, spatialization refers to the distribution of sound in an acoustic space, often in a dynamic way. This practice has been an integral part of experimental electronic music since its early days. The spatialization of previously produced tape music on multichannel loudspeaker setups, known as *Diffusion*, has already been practiced by Pierre Schaeffer in the 1950s. Since then, various techniques have been developed, which are applied in conventional music production, movie sound, video games, XR environments and experimental contexts.

The remainder of this section introduces several concepts and related work relevant to the proposed approach. Section II explains the basic concept of autogenous spatialization and the single components of the toolbox, followed by details on the implementation in Section III. Use cases and a conclusion are presented in Sections IV and V, respectively.

A. Object-Based Spatialization

Object-based spatialization emulates the natural behavior of sound sources by generating the impression of sound coming from specific directions or locations. Also referred to as *mimetic spatialization* [1], this approach is widely used in music production, video games, XR applications and movie sound. Typically, such spatial rendering techniques work with the point source paradigm, in which a sound source is assigned a virtual position in the listening space, for example using Cartesian or spherical coordinates.

Object-based spatialization can be realized with panning algorithms like Vector Base Amplitude Panning (VBAP) and sound field synthesis methods like Ambisonics and Wave Field Synthesis (WFS). These methods are usually agnostic of the loudspeaker system, assuming it to be as dense, evenly distributed and homogeneous as possible, to create realistic and immersive listening experiences. As a consequence, object-based compositions are highly portable. Audio material and source movement data can be stored separately and rendered to any loudspeaker setup matching the requirements. Although the individual experience and quality strongly depend on the loudspeaker systems, they can be regarded as replaceable technical infrastructure.

B. Loudspeakers as Musical Instruments

Unlike object-based methods, channel-based spatialization does not rely on virtual sound positions. These approaches are less suitable for mimicking physical sound sources, but offer enhanced possibilities for sound design. In general, channel-based spatialization allows loudspeakers and individual loudspeaker setups to be used as musical instruments.

In the field of Electroacoustic Music, especially in Musique Contrète and acousmatic music, the individual loudspeaker and loudspeaker setups have long been considered a musical instrument or part of a composition. Iannis Xenakis' tape composition *Poème Électronique* for the Phillips Pavilion at the 1958 Brussels World Fair [2] defined loudspeaker trajectories as part of the architecture, to let sound move along specific paths. Luigi Nono's Prometeo [3], premiered in 1984, uses such trajectories for spatialization in a live, mixed music context. The *Halaphon*, a hybrid analog-digital spatialization system, could route input signals to the speakers during the performance.

Loudspeaker orchestras, most notably the *Acousmonium* [4], combine loudspeakers with different characteristics and exploit these differences in the live Diffusion of tape music. This tradition has been continued by modern versions, as for example the BEAST [5].

ISBN: 979-8-3503-1104-4/23/©2023 IEEE

Spherical loudspeaker arrays, such as the IKO [6], rely on reflections to distribute sound in space. In doing so, the room itself becomes part of the instrument.

Garcia et al. present tools for channel-based spatialization with the OM-SoX library in OpenMusic [7]. An example of spectral spatialization distributes audio signals to a fixed set of loudspeakers, based on a filter bank. Each loudspeaker of an eight-channel setup is assigned a band-pass filter. Thus, the signal shifts between the speakers, according to its spectral content.

C. Spatial Sound Synthesis

In spatial sound synthesis, the spatial aspects are considered at an early stage of the synthesis process and are thus an integral part of the concept. Most established sound synthesis methods provide means for implementing them with consideration of a spatial component. Often, this is a combination of these methods with point sources in the object-based paradigm, leading to a *decorrelation* [8] of the sounds' components.

1) Object-Based Methods: Spatio-Operational-Synthesis SOS [9] rotates single partials of basic waveforms in an additive synthesis process on a circular loudspeaker setup using VBAP. Sounds of musical instruments can be distributed as point clouds with Spectro-Spatial Sound Synthesis [10]. Granular techniques have been used with spatialization methods, based on swarm movements [11] or dictionary-based methods [12]. Einbond and Schwarz [13] propose a spatialized approach to concatenative synthesis for timbre spatialization.

Similar to generic audio effects, spatial audio effects can be applied with extreme parameter settings, alienating the original signal beyond recognition and thus turning them into methods for sound synthesis. Basic operations like amplitude modulation and frequency modulation, if applied with high rates and modulation depths, become synthesis techniques, as for example in FM synthesis. In Spatial Modulation Synthesis [14], the same principle is utilized in the spatial domain. Signals are shifted between loudspeakers at a high rate, resulting in dominant amplitude modulations and Doppler shifts. This is implemented using panning and point-source paradigms, such as stereo, DBAP, VBAP, Ambisonics techniques. The result is considered a *complete instrument, unifying space and timbre* [14].

2) Channel-Based Sound Synthesis: More abstract methods for spatial sound synthesis rely on channel-based paradigms. In *Topographic Synthesis* [8], each loudspeaker of a multichannel system is assigned an individual instance of a synthesis process. This can be any general synthesis technique, such as additive, FM or granular synthesis. When these parallel processes are deterministic and driven with the same input parameters, all speakers' signals are identical. Parameter distributions can be used to create instantaneous or evolutive *spatial textures*. Topographic Synthesis is agnostic to spatial layouts of loudspeaker systems, treating the loudspeakers as sorted arrays.

II. ZERR∗ CONCEPT

`Zerr⋆` is a channel-based spatialization concept for arbitrary audio sources and loudspeaker setups. It distributes audio signals to multiple loudspeakers, based only on the signals' inherent features as well as the properties of the speaker array setup. Although various aspects of the algorithm can be tuned, also during run-time, the audio signal itself defines the spatial distribution. Depending on the parametrization, the approach can alter the original signal significantly, similar to modulation and distortion effects. The interplay of the input signal, algorithm and loudspeaker configuration is considered the instrument, allowing performers to shape the sound in texture, timbre and spatial behaviour simultaneously. The `Zerr⋆` approach is thus not intended as a *mimetic spatialization* tool but aims at a deep coupling of signal features with spatial behaviour.

Figure 1 shows the general signal flow of the `Zerr⋆` approach. It generates N individual signals x^* for N loudspeakers from a mono input signal x. All solid paths represent audio rate signals with the bold paths being the raw input and audible output.

A. Feature Tracker and Feature Processor

In the first stage, audio features are calculated from a mono input signal. Implementations of the included audio features are based on the descriptions by Lerch [15]. Tables I and II list standard time-domain and spectral-domain features that have been implemented to this point.

TABLE I: Temporal features.

Time Domain	
	Root mean square
	Zero crossing rate
	Crest factor

TABLE II: Spectral features.

Spectral Domain	
	Spectral flux
	Spectral centroid
	Spectral rolloff
	Spectral flatness

Alongside the typical instantaneous features, we extract event-related features to function as trigger signals. These event-related features can be inherent to the audio, like zero crossings, or they may be derived from various trajectories of either the source audio or the instantaneous features. All implemented transformations are listed in Table III. Given the modular design of `Zerr⋆`, extracted features can be transformed and combined using arbitrary operations, especially when used in environments like Pure Data.

Derived from German *Zerräumlichung* ≈ spatial disintegration.

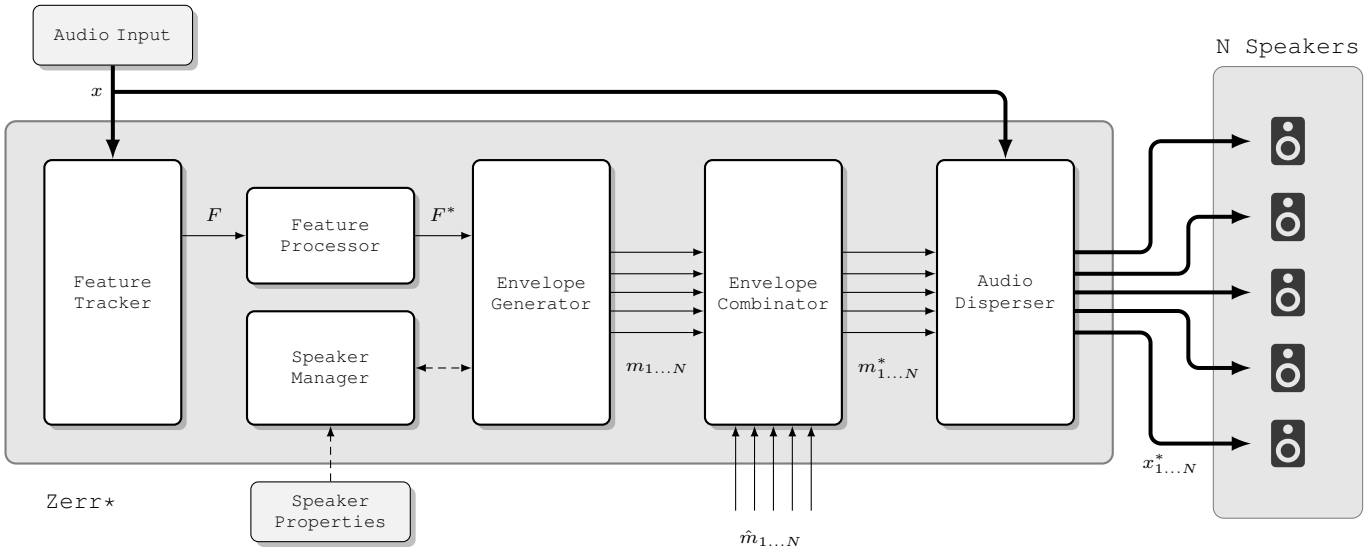

Fig. 1: General flow chart of the Zerr⋆ approach for $N = 5$ loudspeakers.

TABLE III: Trigger transformations.

Trigger Transformations	
	Zero crossing
	Local extrema
	Inflection points
	Threshold violations

TABLE IV: Geometric loudspeaker properties.

Position (Cartesian)	x
	y
	z
Position (spherical)	azimuth
	elevation
	distance
Orientation	yaw
	pitch

B. Speaker Manager

The Speaker Manager is responsible for managing the properties of speakers within a given speaker array configuration. It provides various functions for querying speaker properties, selecting specific speakers, and sorting speakers according to their properties, all of which are essential for the Envelope Generator stage. The Speaker Manager holds standard properties that are defined during initialization, as well as additional specific properties that can be assigned during runtime.

1) Standard Properties: The standard properties include a unique index number and the geometry of the speaker array like the position in Cartesian and spherical coordinates and the orientation of the loudspeaker, as listed in Table IV. Since spherical coordinates and Cartesian coordinates have individual benefits and can be applied for different purposes, both can be used in the Envelope Generator stage. However, only one format needs to be defined, while the other is calculated automatically after loading the configuration. The orientation system includes the two degrees of freedom yaw and pitch.

2) Specific Properties: Specific properties can be defined manually or calculated algorithmically. These properties are function-related rather than inherent and are utilized to achieve specific spatialization.

a) Speaker Masks: In large-scale speaker array setups it can be advantageous to utilize only a subset of speakers in a single Zerr⋆ patch. This enables parallel computation across multiple software instances or machines. The speaker mask is used to determine the visibility of each speaker, where only unmasked speakers are included in the Zerr⋆ system.

b) Speaker Trajectory: Speakers in Zerr⋆ can be interconnected sequentially to establish a trajectory that corresponds to the arrangement of speaker positions. Consequently, sound can traverse along this trajectory. This arrangement can potentially function as criteria for algorithmically sorting the selected speakers, such as by height or azimuth. Alternatively, there is option to manually define the trajectory, allowing for the creation of a customized design that yields a unique path for sound movement.

c) Speaker Topology: Beyond geometry, speaker topology is defined by assigning each speaker a list of indexes corresponding to other speakers linked to it. These can be bidirectional or unidirectional, allowing for customized logical connections between loudspeakers. Defining topology enables more flexible mapping strategies beyond geometric constraints, catering to unconventional speaker setups.

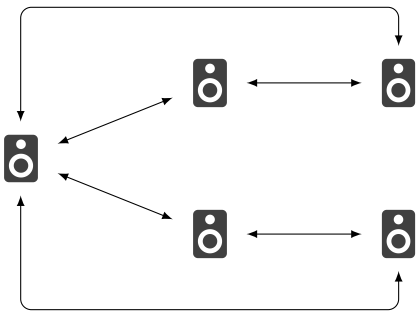

Fig. 2: Example for a loudspeaker topology.

C. Envelope Generator

The Envelope Generator creates N individual modulation signals m_n based on different paradigms. It receives instantaneous and trigger signals from the feature processor. The outputs of the Envelope Generator are scaled between 0.0 and 1.0 and can thus be used as universal envelope signals. The input signals primarily control the Envelope Generator in two main aspects: speaker selection and distribution processing.

1) Speaker Selection: `Zerr⋆` introduces two approaches to speaker selection: Trajectory Mapping and Trigger Shifting.

a) Trajectory Mapping: In the Trajectory Mapping approach, the Envelope Generator takes a processed control signal F^*, ranging from 0 to 1, and maps it to the properties of a speaker layout. This mapping process is illustrated in Figure 3, where the input trajectory is mapped to eight speakers connected sequentially. During the time interval from T_1 to T_2, the signal is shifted from speaker 4 to speaker 5.

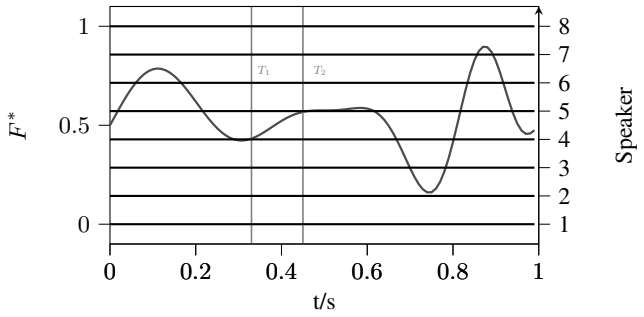

Fig. 3: Mapping a scaled feature trajectory F^* to 8 speakers.

The method used to interpolate between speakers can vary, ranging from linear interpolation to hard threshold or other interpolation techniques, as depicted in Figure 4. The choice of interpolation method can significantly influence the resulting audio experience and the smoothness of transitions between speakers.

b) Trigger Shifting: In the trigger shifting approach, whenever a trigger input is received, the envelope target instantaneously shifts to a newly selected speaker. The Speaker Manager determines the jump destinations by combining

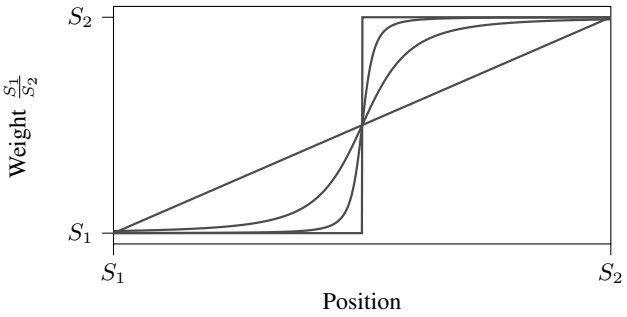

Fig. 4: Interpolation between two neighboring speakers.

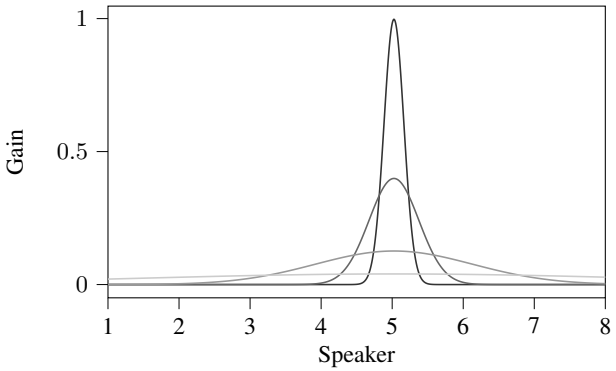

Fig. 5: Spread parameter for shaping the distribution.

speaker topology and geometry. two methods are provided for decision-making:

- **Random:** The next jump destinations are randomly selected from the available candidate speakers, introducing an element of unpredictability to the speaker selection.
- **Nearest:** The next jump destinations are selected based on which candidate speakers are closest, in terms of geometry or topology, to the currently mapped speaker.

2) Distribution Processing: The distribution created in the Envelope Generator can be shaped by changing the spread and the overall gain.

a) Spread: Input sound can be spread to all other speakers in addition to the central one with lower gain to create a more immersive sound field. The relative volumes of the nearby speakers are computed using a normal distribution and the distance to the main speaker, as depicted in Figure 5. Eight speakers are equal distance distributed in which speaker 5 is the central speaker to this point. The extent of sound spread is controlled by altering the variance.

b) Overall Gain: The overall gain can also be modulated. When connected to a slowly varying signal, it introduces additional details in the sound texture. When the trajectory varies rapidly, it can resemble amplitude modulation and result in a significant alteration of the original sound.

D. Envelope Combinator

The envelopes created by the Envelope Generator serve as modulation signals for the original audio input. Two modules

are proposed for further processing the envelope and combining it with the original audio input. The Envelope Combinator provides functions to easily combine sets of envelopes from different Envelope Generators. In addition to basic numerical calculations (e.g. summing, averaging, maximizing), K sets of envelopes can be combined as follows:

$$m_n^* = \sqrt[k]{|\prod_{i=1}^{k} m_{(n,i)}|} \tag{1}$$

This functionality is useful when users desire to utilize more complex envelope generation strategies, for example in two-dimensional panning. It's worth noting that this module is optional since a single Envelope Generator already provides sufficient information for dispersing the original audio.

E. Audio Disperser

The Audio Disperser module generates the individual loudspeaker signals x_n^* by applying the corresponding modulation signal m_n on x:

$$x_n^* = x m_n^* \tag{2}$$

The original audio signal is modulated by the corresponding generated envelopes and distributed to the speakers.

III. Implementation Details

A. Module Design

The `Zerr*` system is implemented in C++ using a modular design to enhance extendability and reusability. The architecture of the `Zerr*` system aligns with the module segmentation diagram shown in Figure 1. Each module, with the exception of the Speaker Manager, is implemented as an individual signal processing unit whose I/O signals operate exclusively at an audio rate. The Speaker Manager is a sub-module within the Envelope Generator. Operating at an audio rate allows the system to be encapsulated in various ways, which will be elaborated upon in the subsequent section.

The design logic of the feature tracker module draws inspiration from the Essentia library. Each audio feature extraction algorithm is implemented using a consistent class template and is accessed through a uniform calling interface within the feature tracker module. This uniformity simplifies the process of adding new audio feature algorithms to the system. Homogeneous processes, such as audio buffering and Fourier transforms, are conducted in the main call module to prevent redundant calculations.

Furthermore, the modular design grants users the freedom to leverage the system in unconventional manners. For instance, they could singularly employ the Feature Tracker and devise their own mappings or use self-generated signals to drive the envelope generation process. The design imposes minimal constraints on user operations.

https://essentia.upf.edu/

B. Encapsulation

The repository provides two encapsulations of the `Zerr*` system: as a standalone JACK client and as Pure Data objects. In the JACK implementation, all modules are internally buffered and connected according to the standard signal flow, forming a cohesive system signal processing unit. The JACK audio client essentially encapsulates the system's overall inputs and outputs. For the Pure Data objects, each module operates as an independent processing unit. The signal I/O and configuration interfaces are initially encapsulated as C data structures, aligning with the Pure Data design concepts. These are subsequently compiled as independent objects using the PD API. Inter-module signaling is managed by Pure Data.

Each module offers interfaces for reading system parameters from YAML files and, at a more granular level, for directly modifying system parameters. For the JACK client, module parameters are read exclusively from the configuration file during system initialization. The Pure Data objects, however, offer methods for modifying the configuration using object arguments or by message dispatch, enabling users to adjust settings in real-time. There are two types of configuration files available: the speaker array configuration and the module configuration. The former covers the standard properties elaborated upon in the Speaker Manager section. The latter houses configurations for all modules, like the chosen features and the speaker mapping strategy etc.

C. PD Package

The following objects are part of the Pure Data version:

- `zerr_features~`
- `zerr_envelopes~`
- `zerr_disperser~`
- `zerr_combinator~`

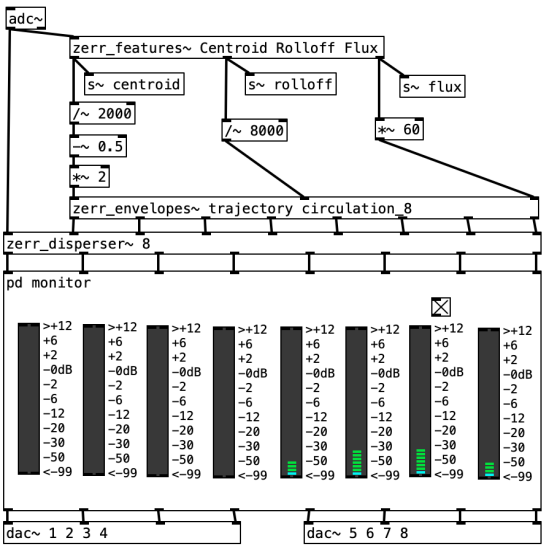

Fig. 6: Example PD patch for eight loudspeakers.

https://github.com/ringbuffer-org/Zerr
https://jackaudio.org/

Figure 6 shows an example patch that contains all necessary `Zerr*` objects. A separate Feature Processor module is not included because its functionalities can be readily achieved using built-in Pure Data objects, rendering an additional module superfluous. Dataflow programming languages like Pure Data are best suited for exploring different combinations and interconnections of the basic building blocks and allow additional manipulations with built-in processing units.

IV. Tests and Use Cases

During the first experimental stage, the `Zerr*` approach has been applied as a standalone JACK client during various *QuarantineSessions*. Since these online concerts distribute binaural streams, a virtual loudspeaker setup was used in Ambisonics to allow spatialization in two dimensions. Figure 7 shows an example for a virtual eight-channel setup, arranged as a circle around the listener. This virtual setup was used with different eight-channel `Zerr*` algorithms, using the properties of the virtual loudspeakers.

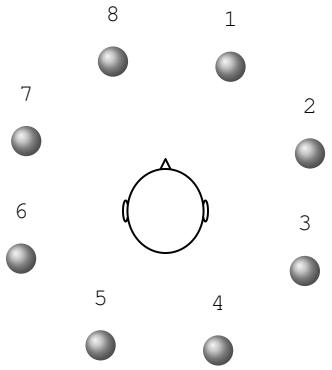

Fig. 7: Virtual eight-channel setup for binaural rendering.

The Pure Data version of the `Zerr*` system has been tested in studio sessions, using 21 loudspeakers in a default dome configuration and selected sub-sets of the loudspeakers.

In both test scenarios, the spatialization approach was more appropriate for simple synthetic signals, such as combinations of filtered noise and basic waveforms. These signals have a direct connection to the extracted audio features and thus make the control comprehensible and more immediate. In addition, strong modulations enrich these simple signals, whereas recorded sounds can lead to less appealing effects and disturbing artifacts.

V. Conclusion

In its recent state, the proposed approach for autogenous spatialization can be used as a tool for composition and performance. Informal tests have shown the capabilities of the tools for binaural synthesis and a standard a loudspeaker dome. Working at the boundaries of spatialization and spatial sound synthesis, the approach is most suited for modular synthesizers and music programming environments.

A visual programming language like Pure Data seems ideal for combining the `Zerr*` objects in different ways and with additional objects, thus exploring the possibilities. Additional music programming environments will be included as targets to allow a widespread use. Further steps also aim at a user study with a variety of electronic and traditional instruments, as well as arbitrary loudspeaker configurations.

Acknowledgements

Thanks to Miller Puckette for his advice and input for the implementation in Pure Data.

References

[1] K. L. Hagan, "Textural composition: Aesthetics, techniques, and spatialization for high-density loudspeaker arrays," *Computer Music Journal*, vol. 41, no. 1, pp. 34–45, 2017.

[2] V. Lombardo, A. Valle, J. Fitch, K. Tazelaar, S. Weinzierl, and W. Borczyk, "A virtual-reality reconstruction of poeme electronique based on philological research," *Computer Music Journal*, vol. 33, no. 2, pp. 24–47, 2009.

[3] M. Brech and H. von Coler, "Aspects of space in Luigi Nono's Prometeo and the use of the Halaphon," in *Compositions for Audible Space*, ser. Music and Sound Culture, M. Brech and R. Paland, Eds. transctript, 2015, pp. 193–204.

[4] S. Desantos, C. Roads, and F. Bayle, "Acousmatic morphology: an interview with François Bayle," *Computer music journal*, pp. 11–19, 1997.

[5] S. Wilson and J. Harrison, "Rethinking the BEAST: Recent developments in multichannel composition at birmingham electroacoustic sound theatre," *Organised Sound*, vol. 15, no. 3, pp. 239–250, 2010.

[6] F. Zotter, M. Zaunschirm, M. Frank, and M. Kronlachner, "A beamformer to play with wall reflections: The icosahedral loudspeaker," *Computer Music Journal*, vol. 41, no. 3, pp. 50–68, 2017.

[7] J. Garcia, J. Bresson, M. Schumacher, T. Carpentier, and X. Favory, "Tools and applications for interactive-algorithmic control of sound spatialization in openmusic," in *inSONIC2015, Aesthetics of Spatial Audio in Sound, Music and Sound Art*, 2015.

[8] E. Nyström, "Topographic synthesis: Parameter distribution in spatial texture," in *Proceedings of the International Computer Music Conference (ICMC)*. ICMC, 2018, pp. 117–122.

[9] D. Topper, M. Burtner, and S. Serafin, "Spatio-operational spectral (SOS) synthesis," in *Proceedings of the International Conference of Digital Audio Effects (DAFx)*, Singapore, 2002.

[10] H. von Coler, "A JACK-based application for spectro-spatial additive synthesis," in *Proceedings of the 17th Linux Audio Conference (LAC-19)*, 2019.

[11] S. Wilson, "Spatial swarm granulation," in *Proceedings of the International Computer Music Conference (ICMC)*, 2008.

[12] A. McLeran, C. Roads, B. L. Sturm, and J. J. Shynk, "Granular sound spatialization using dictionary-based methods," in *Proceedings of the 5th Sound and Music Computing Conference (SMC)*, 2008.

[13] A. Einbond and D. Schwarz, "Spatializing timbre with corpus-based concatenative synthesis," in *Proceedings of the International Computer Music Conference (ICMC)*, New York, United States, Jun. 2010, pp. 1–1.

[14] R. McGee, "Spatial modulation synthesis," in *Proceedings of the International Computer Music Conference (ICMC)*, 2015.

[15] A. Lerch, *An Introduction to Audio Content Analysis: Applications in Signal Processing and Music Informatics*. John Wiley & Sons, 2012.

https://ccrma.stanford.edu/search/node/QuarantineSession