An Algorithm for Emulating Stereophonic Microphone Arrays

Jeffrey M. Clark, M.M.
Music Theory and Composition
School of Music, Ball State University

***Abstract*—This paper outlines an algorithm an virtual studio technology plug-in for emulating the effect of a stereophonic recording array on a monophonically recorded sound source. This emulation will localize the sound within the perceptual soundstage using the time-domain and level-domain cues appropriate to the microphone array specifications chosen by the user. The document covers the theoretical basis for the algorithm, as well as the design decisions made during the specific implementation.**

The audio plug-in specified in this document was designed as an entry to the 2020 AES/MATLAB plug-in design competition.

***Index Terms*—Audio Signal Processing, Digital Audio, Stereophonic Localization, Panning, VST, Plug-in, AES, MATLAB**

I. Overview

AUDIO localization in stereophonic recording focuses on balancing the pressure level and time difference cues that inform audio localization through interaural time differences (ITD) and interaural level differences (ILD). Within the stereophonic recording praxis, it is understood that the weighting of ITD and ILD cues contributes to varying desirable qualities within the resultant soundfield. Stereophonic microphone arrays are also chosen to account for the desired balance of direct sound and reflected sound and to accommodate the recording angle of the perceptual ”soundstage” and translate it to audio reproduction systems with minimal angular distortion.

This is, however, at odds with typical stereophonic localization practice within the application of audio signal processors; which tend to prioritize localization solely through ILD cues. In certain contexts this can lead to a situation where recorded signals that are being mixed together during post-production can be localized using different perceptual locative cues, yet with the intent of producing a coherent, and qualitatively consistent perceptual soundstage.

By modelling the time and level relationships between a sound source and the microphones in a stereophonic recording array, a monophonically recorded sound can be localized within the perceptual soundstage using the time and level cues appropriate to the modeled array. This paper will outline an algorithm and discuss and implementation for abstracting these relationships in to a CPU-efficient model, and exposing the appropriate parameters to the user.

A. A Review of Localization in Stereo Recording

Within the context of localization, it is understood that a relative dominance of time-domain cues creates a greater perceived sense of stereophonic width/envelopment, whereas a greater relative preponderance of level-based cues increases the sense of locative precision. Microphones, as pressure transducers, encode local changes in atmospheric pressure into changes in electrical pressure, and can be practically thought of as sampled points in space. Additionally, microphones have varying types of directivity – represented by a polar plot – that represent how efficiently they transduce sound based on the sound-wave angle of arrival relative to the microphone’s orientation. For directional microphones with cardioid-style polar patterns, the amount of attenuation tends to increase as the sound-wave’s direction of arrival moves further away from the oriented ”front” of the microphone[1].

By positioning two directional microphones in the same location and facing in two different directions, any sound-wave captured by them will be increasingly less attenuated as it approaches the front of one microphone and more attenuated as it approaches the other. If the sound wave approaches from an angle that equiangular to both microphones, then it will be equally attenuated. This equal attenuation, during reproduction, will have the perceptual effect of placing the sound at the center of the virtual sound-stage.

Similarly, by positioning two nondirectional microphones in to locations within the space, the distance between them will create differences in the time of arrival based on the speed of propagation of the sound-wave (the speed of sound). As the direction of arrival shifts away from being perpendicular to an imaginary line drawn from one microphone to the other, there will be an increasing time delay between when the sound is captured in the closer microphone and when it is captured in the further microphone. As with the level-based cues, when

[1]There is some complexity with this as the directivity pattern value increases past a pure-cardioid, with an inverse-phase area beginning to present at the rear of the microphone and increasing until the directivity approaches a bidirectional pattern

the time of arrival is equal (i.e. there is no delay in one microphone) then the perceptual effect during reproduction is one of the sound being centered in the perceptual soundstage. As the delay increases in one microphone, the sound will appear to come from the direction in which it arrives first[2].

These level and time-based principles can be freely combined, creating a spectrum of options for recording engineers to choose from. More advanced methodologies will add in a center microphone (such as the popular "Decca-Tree" configuration), and/or will also add flanking/outrigger microphones as well. The combined effect of these various microphone set-ups yields a complex interplay of time and level cues.

B. Signal-Processing Panning Methods

Current methods in stereophonic panning within digital audio workstations (DAW) typically focus on the manipulation of level differences. These level differences tend to follow either a sine-cosine (see 1) or a linear curve (see 2).

For $\{p \mid 0 \leq p \leq 1\}$ to represent the range of the user input value for stereophonic panning, then the normal panning functions can be found as:

$$\begin{bmatrix} L \\ R \end{bmatrix} = x * \begin{bmatrix} \cos \frac{p\pi}{2} \\ \sin \frac{p\pi}{2} \end{bmatrix} \tag{1}$$

$$\begin{bmatrix} L \\ R \end{bmatrix} = x * \begin{bmatrix} p \\ 1-p \end{bmatrix} \tag{2}$$

Interestingly, there is not as accepted standard functionality for DAWs to perform panning based on time-domain cues. Multiple reasons can be found for this, though the two that arguably stand out are: lack of monophonic compatibility due to phase distortions, and tradition from analogue mixing consoles which were unable to provide the delay lines necessary. Additionally, the usage of computer memory for the circular-buffer delay-lines needed may have also been a consideration against memory availability in early computers.

The use of time-domain cue is either implemented through a stereo "widener[3]" or by manual implementation through simple or purpose-built delay audio plug-ins in a technique known as the "Haas Trick[4]."

[2]This phenomenon is known as the "precedence effect"

[3]The implementation of stereo widening techniques varies from processor to processor, not all will use the same methods. Any given widener may not actually use time-based cues, relying instead solely on frequency-domain adjustments or dynamics range processing.

[4]after Helmut Haas, who studied the psychoacoustical implications of the precedence effect in his Ph.D. thesis.

C. Purpose

This paper will describe an algorithm for modeling the interplay of the microphones within a stereophonic recording array; and discuss an audio plug-in that implements this algorithm. This algorithm and the accompanying audio processor were developed as an entry to the AES/MATLAB Plugin Competition in 2020. As such, the processor developed was designed to work within the limits of the competition – most specifically the two-channel input/output requirement, and the goal to solve a specific use-case scenario. Theoretically, this algorithm could be applied to any arbitrary microphone set-up and source placement within a virtual space.

The immediate use-cases intended to be addressed with this specific implementation are: 1. the placement of monophonically recorded sources into a stereophonic microphone array, and 2. the addition of time-domain cues through a UI that is intuitive and effective, and yields natural and predictable results.

1) Blending monophonic encoded sources: When recording acoustic ensembles, recording engineers will frequently employ both a main stereo array and area/spot microphones. These spot microphones are used to accentuate various parts of the ensemble to difference logistical and aesthetic ends. Drawing from previous sections, stereo recording praxis and panning implementation in DAWs do not agree on how to encode stereophonic localization cues. This can quickly lead to a situation where the sound of an instrument recorded in an ensemble with both a stereo array and spot microphone will have a mixture of time/level cues in the main array signal, but only be localized with level cues in the panned signal from the spot microphone. Due to the differences in the perceptual effect of time and level cues, this can lead to inconsistencies in the quality of the soundfield as the dominance of the source of the instrument in the audio mix changes between the main array and the spot microphones; this may also cause inconsistency in instruments that are around the spot microphone. The use of this processor will encode the monophonic sound with time and level cues that closely match that of the main array, meaning that its localization and sense of envelopment will not change based on the dominance of the source.

Following this, another context for this application of the processor is in blending a monophonic recorded source into a recording that was taken of an ensemble using a stereophonic microphone array. "Distributed recording" is not uncommon – especially in music for media. It is not uncommon for sections of an ensemble to be recorded independently and then put together during post-production. This processor would allow for independently recorded instruments to be better blended into a larger ensemble by encoding them with the same localization cues, creating a greater qualitative unity in the resultant soundfield.

This also has implications for sample-based music production and the common practice of laying samples of the same instrument family that come from different sample libraries. Commercial sample libraries each have their own recording and staging procedures, and different instruments and sections can be localized using different combinations of level and time-domain cues. The use of this processor on the spot microphone or monophonic summed options will allow for the sounds to be spatialized together in a more consistent manner.

Finally, this processor can also be used in conjunction with standard post-processing artificial reverberation techniques to manufacture a sense of unity from completely independently recorded instruments by encoding them onto a virtual soundstage using natural level and time-domain cues.

2) Time-domain cues in panning: A secondary use for this processor is in presenting an intuitive method of adding time-domain cues into panning. Rather than relying on an arbitrary implementation of the Haas trick, a user of this processor can – through the familiar metaphor of a stereophonic microphone pair – creating a consistent, repeatable, and intuitive panning plan for components of their audio mixdown that includes both time and level perceptual components.

II. Definitions

A. The Virtual Soundstage

The abstracted virtual space will be a two-dimensional space with orthogonal basis $(\hat{x}, \hat{y})$. For ease of visualization, $\hat{x}$ will be considered as left-pointing, and $\hat{y}$ will be considered as pointing forward and bisecting the soundstage. Thus, the used virtual soundstage area will be the area within the range of positive y; the positive x direction will be perceptual left, and the negative x direction will be perceptual right.

B. The Sound Source

1) The propagation of sound from real sound sources: The first immediate barrier to modeling the propagation of sound from a sound source is the nature of any given sound source. Few acoustic resonators approach being perfect point sources with equal spherical radiation of sound across the entire audible frequency spectrum. It is, in theory, possible to model the emission of sound using frequency-dependent statistical directivity data derived from various instruments. This model could be applied to the algorithm if the facing of the sound source was known. However, chasing down this trail raises questions into the propagation of sound towards any given microphone, the effect of the physical characteristics of the sound stage, and the effect of height components in the positioning of the source and microphones.

Trying to account for all of these components would eventually necessitate venturing into the realm of physical modeling. While such an undertaking is a worthwhile avenue of research, it breaks this processor's goals of ease of use and intuition of abstraction, as well violating the goal of remaining computationally efficient enough to use as a general DAW plug-in.

A review of literature regarding stereophonic recording practice tends to focus on the directional characteristics of sound sources as considerations for the height of microphones in the array, with the aim of obtaining the desired blend of each source's characteristic frequency components, as well as informing the optimal placement of spot/area microphones. The effect that a sound source's directivity can have on localization does not seem to be a remarkable characteristic of stereophonic recording arrays.

2) The virtual sound source: Based on the typical assumptions of stereophonic recording praxis, the virtual sound source can be considered as an ideal sound radiator which radiates sound equally towards every measured virtual point. The virtual sound source, S, has its location at (r, θ) and does not have any other properties – it does not inherently apply any frequency/amplitude transfer function onto the pre-encoded signal.

The useable range of θ_S, in keeping with the bounds of the defined virtual soundstage is $[0, \pi]$. The radial distance r_S is an abstraction of a real distance in meters.

Since the virtual sound source only imparts positional data to the sound, the actual recorded sound to be encoded can also be represented by S; where S is the recorded sound, which is emitted into the virtual space at point (r, θ)[5]

C. Virtual Microphones

The virtual stereophonic array can be represented as a set of virtual microphones, $M \mid M = \{m_1, ...m_n\}$. Each virtual microphone has its placement within the virtual space at coordinates (x, y), and has orientation θ, with polar directivity factor p.

The orientation, θ of the virtual microphone represents is direction of facing into the soundstage. Like θ_s, the range of usable values is $[0, \pi]$ where a rotation of 0 indicates a facing directly to the left, a facing of π is directly to the right, and an orientation of $\frac{\pi}{2}$ points the virtual microphone directly forward – parallel to the y-axis with a lateral offset of x_m.

[5]The choice of polar coordinates to represent the position of S is semi-arbitrary, and does owe some part to the specifics of the implementation. An origin shift will be necessary during the processing which will require the translation of the polar coordinates into cartesian, however there will be a second translation back to polar coordinates again. In this sense, the source is always conceptually considered based on its angle and distance from some reference point. Thus, it seems natural to define S in those terms.

The polar directivity value, p is a real number in the range $[0, 1]$ where $p = 0$ yields a non-directional polar directivity pattern, and $p = 1$ yields a bidirectional polar directivity pattern. The resulting polar attenuation pattern at angle of arrival (relative to the "font" of the virtual microphone) ϕ for any value of p can be found as:

$$m(p, \phi) = (1 - p) + p\cos(\phi) \tag{3}$$

1) Limitations in abstracting ideal microphones: It should be noted that real microphones' polar patterns are frequency-dependent, with cardioid microphones ($0 < p < 1$) becoming increasingly less directional at lower frequencies. At higher frequencies, the physical body of the microphone and the diameter of the microphone diaphragm also begin to affect in the polar response. Additionally, cardioid microphones – due to the physical mechanics of generating a cardioid polar pattern – tend to exhibit a marked attenuation in their frequency response at lower frequencies.

Often, the specific color that a specific model of microphone imparts on the recorded sound is carefully chosen, with the choice of microphone being considered one of the major creative decisions in recording and the design of many microphones being intentional to support specific colorations and yield various creative options. It would be impractical to try and model the specific responses of common microphones used in main arrays, and the processing needed to apply the frequency-domain modeling would break the CPU-efficiency goal. Additionally, an attempt to reproduce the frequency-domain aspects of a stereophonic array would have the effect of distorting the (likely intentionally-chosen) frequency-domain aspects of the source recording.

Thus, using virtual microphones with ideal polar patterns is likely the optimal use-case for most users. This will be with the understanding that the virtual microphone array will impart localization cues that approach that of a real microphone array, but – due to limitations in modeling real-world physics – will not fully replicate the cues of a real-world microphone array. However, the processor will provide localization cues that are closer than level-difference-only panning methodologies.

There is one case that is worth considering. As previously noted, directional microphones tend to exhibit a marked attenuation at low-frequencies. There are some recording techniques that take advantage of this tendency, especially since the human prioritization of localization cues is frequency-dependent. It may be desirable to include some simple high-pass and low-pass filter options within the implementation to allow the user to emulate some of the key frequency-dependent behavior.

D. The User Interface

In the implementation of the processor, the relevant controls for defining the desired virtual objects and their parameters need to be exposed to the user. For the virtual sound source, this means exposing controls for θ_S and r_S. Then the number of virtual microphones in set M needs to be defined, the relevant values for p_m, (x_m, y_m), and θ_m need to be exposed.

1) Orienting the user within the virtual space: The first concern is presenting the orientation of the virtual space to the user in a way allows them to be intuitively oriented within and able to intuitively understand the distances and placements of virtual objects with in. Thus, establishing an intuitive reference for the placement of the origin of the space is of primary importance.

Following the specific facings of the basis $(\hat{x}, \hat{y})$, the most logical place to orient O would be at the central point of the microphone array – which is also what user will intuit as the "perceptual center". This also gives a consistent and predictable basis for user to take real-world measurements and translate them into the virtual abstraction used by the processor.

2) Orienting the virtual sound source: The typical mixing engineer is likely to naturally think of angular rotation in terms of radians. Similarly, considering the amount of angular rotation from the x-axis is not reflective of the "front" oriented human experience. Thus, the user interface is better presented with θ_S translated as degrees of deflection from forward. Additionally, there is a bias (at least within Western culture) that a numerical increase in amount be translated as right-ward motion. Due to this, the number of degrees needs to decrease with counter-clockwise rotation.

Thus, if ϕ is the value, in degrees of deflection from forward, then:

$$\theta_S = (-\phi + 90) * \frac{\pi}{180} \mid \phi = [-90, 90] \tag{4}$$

Since r is abstracted as meters, the user can specify the distance r from point O directly – understanding it as the distance from the center of the real (or imagined) microphone array.

3) Constructing the microphone array: The set of microphones in the virtual array, M, is some number of virtual microphones. While it is possible to define an arbitrary number of virtual microphones each at any conceivable position within the virtual space and define the algorithm in a way to accommodate, usual stereophonic recording practice, again, gives some practical limits that can be used to refine and focus the amount of controls exposed to the user and enhance the quality of the user experience.

The fundamental praxis of stereophonic recording is based on the use of a microphone pair, that translate to the raw left and right audio channels. This microphone pair is laterally spaced at some distance, and has an angular splay of some angle. This primary pair can be designated the "main' stereo pair, m_{mains}, that has the properties of its lateral distance, d, angular splay, ϕ.

As discussed in the overview, it is also common to use a second pair of microphones in tandem to the first, that occupy a wider spacing. These additional microphones can be designated as the "flanks", m_{flanks}. The flank pair will have the same properties, d and ϕ, as the main pair.

Finally, there is a long tradition of using a center microphone, m_{center}. Typically, the center microphone does not rotate in the 2D plane; so, it will only have a distance, d. However, for a center microphone, the displacement is forward, rather than lateral; following this, any value for d_{center} will be long the y axis.

This gives a maximum size of M of $n = 5$, which is a manageable number. These are presented to the user as two pairs and the single center microphone. The preferred ordering of the microphones within M will follow the order of expected frequency of use (mains, flanks, center), and the audio-industry preference for left-right ordering of channels:

$$M = \{m_{\text{mL}}, m_{\text{mR}}, m_{\text{fL}}, m_{\text{fR}}, m_{\text{c}}\} \tag{5}$$

The two pairs and center microphone are each switchable, so that the user can freely enable/disable any combination of them. Each microphone group also has a control exposed that applies an amplitude scalar to the signal of that group, to allow for different proportions of that group's signal into the output[6]. This scalar, g, has an adjustment range in decibels: $g = [-20, 0]$.

4) Identifying virtual microphone distance limits: Identifying practical limits for the d value for the two pairs will require a bit of arbitrary demarcation. The main pair must allow for a minimum of $d_{\text{mains}} = 0$ to accommodate coincident microphone techniques. Likewise, for flexibility and simplicity there is little reason not to allow $d_{\text{flanks}} = 0$ (though it would an odd use-case for this value to be used). For the center microphone, while it is common in many cases for it to be displaced forward, there are also a number of cases that require it to remain in-line with adjacent pairs; so a minimum value of $d_{\text{center}} = 0$ is also necessary to accommodate all reasonable use-cases.

For maximal allowed values, and examination of the literature shows that main mic pairs usually occur in spaces smaller than 100cm, however triplets can employ rather wide distances, with "typical" Decca-tree configurations spanning 2m. The current implementation allows for a maximal span of 3m to accommodate some extra working room for three-microphone expanded configurations, additionally, due to the common use of $d < 1m$, the values for d_{mains} are presented to the user in centimeters; this gives $0 \leq d_{\text{mains}} \leq 300$ as the total range for the mains' lateral distance.

After an informal poll and discussion with several sound recordists who specialize in acoustic ensemble recording, an upper limit of 10m was chosen for the flanking pair. This gives $0 \leq d_{\text{flanks}} \leq 10$ as the total range for the flanks' lateral distance, and this presented to the user in meters.

Finally, the maximum amount of forward displacement for the center microphone is was set to $d_{\text{center}} = 100$ in centimeters. Common uses of a forward-set center microphone are under this threshold, and the 100cm point allows for a round number as the limit and offers some extra distance as a buffer to catch reasonable outliers.

5) Microphone UI parameter conversion to values for calculation: A key aspect of many stereophonic recording arrays is bilateral symmetry. Hence the focus on pairs of microphones and defining them in terms of their mutual distance and angular splay. Because of this, it can be assumed that for any of the paired microphones:

$$d_{\vec{mO}} = \frac{d_m}{2} \tag{6}$$

Since the standard abstracted unit of measure in the virtual space is a meter, the distances for the center microphone and the main pair will need to be converted from the centimeter presentation to the user. Since every microphone only moves along one axis, and does not cross the origin, its position along the other axis will remain fixed at 0. For the pairs of microphones, the symmetry at the y-axis means that the left microphone of each pair will remain in the span $d_L \leq 0$ whereas the right microphone in each pair will be within the span $d_R \geq 0$.

Thus, for the pairs of microphones:

$$\begin{bmatrix} x_L \\ x_R \end{bmatrix} = C0.5 \begin{bmatrix} -d_m \\ +d_m \end{bmatrix} \tag{7}$$

Where C is the unit conversion constant: $C_{\text{mains}} = 0.01$ and $C_{\text{flanks}} = 1$.

The value of d_c can be directly converted into meters, and used as the y-coordinate value for the center microphone:

[6]Ideally, the plug-in would be able to output the individual processing of each microphone; however the format of the competition that the plug-in was designed for limits the output channels to two.

$$x_c = 0.01 d_m \tag{8}$$

For the angular splay values of the microphone pairs a similar conversion can be used. Since the angular splay value is the angle between the microphones, the deflection from forward can be found by halving the angular splay. Since perceptual forward is at an angular rotation of $\frac{\pi}{2}$, the left and right microphones' rotation values can be found by adding or subtracting the deflection value (converted to radians) from $\frac{\pi}{2}$:

$$\begin{bmatrix}\theta_L \\ \theta_R\end{bmatrix} = \frac{\phi\pi}{360} J_{2,1} + \begin{bmatrix}+\frac{\pi}{2} \\ -\frac{\pi}{2}\end{bmatrix} \tag{9}$$

Where ϕ is the angular splay value in degrees and J is a matrix of ones.

III. Positional Relationships

A. The Position of the Sound Source

The position of the sound source needs to be referenced both in terms of O and in terms of the position of m. Given θ_s and d_S, its position (x, y) can be found through the usual conversion:

$$\begin{bmatrix}x_S \\ y_S\end{bmatrix} = \begin{bmatrix}\cos\theta_S \\ \sin\theta_S\end{bmatrix} \tag{10}$$

Following this, finding the position of the sound source relative to the microphone, $(x, y)_{Sm}$ can be done by shifting the reference origin in terms of the position of m:

$$\begin{bmatrix}x_S m \\ y_S m\end{bmatrix} = \begin{bmatrix}x_S \\ y_S\end{bmatrix} + \begin{bmatrix}x_m \\ y_m\end{bmatrix} \tag{11}$$

B. Finding $d_{\vec{mS}}$

With the positions of the sound source and the microphones defined in the virtual space, it becomes important to define their relationships to one another. Since the encoding of the source at each microphone depends on the relationship between the source's location and the location of the microphone. These key parameters can be derived from a vector taken from the microphone to the source, $\vec{v}_m = \vec{mS}$.

If follows that:

$$d_{\vec{v}} = \sqrt{(x_s - x_m)^2 + (y_s - y_m)^2} \tag{12}$$

C. Finding θ_{mS}

Since a microphone's polar directivity pattern applies an amplitude scalar based on the amount of angular deflection the source is away from the forward-facing axis of the microphone, $\theta_m S$ should represent this amount of deflection. There are two different methods that can be used to find $\theta_m S$.

1) Difference in rotations: The first method is to find the angular rotation of the position of S around the position of m, and then taking the difference of this angular rotation and the amount of rotation of the virtual microphone (θ_m).

$$\theta_{mS} = \tan^{-1}\left(\frac{y_{Sm}}{x_{Sm}}\right) - \theta_m \tag{13}$$

2) Using linear transformations: Alternatively, the endpoint of $\vec{v}$ could be redefined by applying a rotation matrix based on θ_m

$$\begin{bmatrix}X \\ Y\end{bmatrix} = \begin{bmatrix}\cos\theta_m & -\sin\theta_m \\ \sin\theta_m & \cos\theta_m\end{bmatrix} \cdot \begin{bmatrix}x_{Sm} \\ y_{Sm}\end{bmatrix} \tag{14}$$

$$\theta_{mS} = \tan^{-1}\left(\frac{Y}{X}\right) \tag{15}$$

By combining (14) and (15):

$$\theta_{mS} = \tan^{-1}\left(\frac{x_{Sm}\sin\theta_m + y_{Sm}\cos\theta_m}{x_{Sm}\cos\theta_m - y_{Sm}\sin\theta_m}\right) \tag{16}$$

IV. Virtual Microphone Processing

A. Processing as a Complex Number

The transfer function that a virtual microphone applies to the signal is applied in the time and amplitude domains. This can be represented as a complex number in the form:

$$m(S) = \Delta v + \Delta t \tag{17}$$

Where Δv is the change in amplitude, and Δt is the change in time

B. Δv Calculation

The ΔV component of the function $m(S)$ if based off of the positional relationships of the virtual microphone and the virtual sound source, and the polar attenuation pattern setting of the microphone. The transfer function defined by this relationship can be found as a function of θ_{mS}, $v(\theta_{mS})$. Recalling the polar pattern equation from (3), a virtual microphone can be seen as consisting of two components: a nondirectional component, and a bidirectional component[7]. Following this, by replacing ϕ in (3) with θ_{mS}, the resulting function, $m(\theta_{mS}, p_m)$, can be used to find the amplitude scalar that represents the magnitude of the effect of the virtual microphone's polar attenuation pattern on the source signal, S, when S is at the relative angle represented by θ_{mS}.

Therefore, Δv can be found as the product of the source signal and the angle-dependent scalar coefficient:

$$\Delta v_m = S \cdot m(\theta_{mS}, p_m) \tag{18}$$

Which can rewritten more explicity as:

$$\Delta v_m = S\left[(1 - p_m) + p_m \cos(\theta_{mS})\right] \tag{19}$$

C. Δt Calculation

The Δt component represents the time that it takes for the sound emitted by source S to reach the position of virtual microphone m. Following from (12), the value of $d_{\vec{mS}}$ when combined with the speed of sound constant c can be used to find the time-domain component.

1) Calculating the speed of sound: The speed of sound in real spaces is generally considered to be $343\,\frac{\mathrm{m}}{\mathrm{s}}$. However, it is not a constant. The speed of sound can change based on a number of environmental parameters. To account for this, the speed of sound can be adjusted by some value, Δc, which is implemented in this version as a number $\Delta c = [-10, 10]$; so that:

$$c = 343 + \Delta c \tag{20}$$

Following the usual formula for time:

$$t = \frac{d}{c} \tag{21}$$

[7]This mirrors the construction of some types of real directional microphones.

Substitutions can be made based on (12) and (20) to give the expansion of Δt:

$$\Delta t = \frac{d_{\vec{v}}}{343 + \Delta c} \tag{22}$$

D. $M(S)$ Array processing

It then follows that the complex-number formulation of the function $m(S)$ can be written out:

$$m(S) = S\left[(1 - p_m) + p\cos(\theta_{mS})\right] + \hat{t}\frac{d_{\vec{v}}}{343 + \Delta c} \tag{23}$$

Which is comprised of two components: the adjusted amplitude of S, and the time-domain shift of S.

The virtual microphone function (17) can be applied for each element in M to find the effect of the total microphone array:

$$M(S) = \{m_1(S)...m_n(S)\} \tag{24}$$

The elements of $M(S)$ can be apportioned to the mixdown channels following:

$$\begin{bmatrix} L(M(S)_n) \\ R(M(S)_n) \end{bmatrix} = gM(S)_n \begin{bmatrix} k_L \\ k_R \end{bmatrix} \tag{25}$$

Where g is a scalar constant, and k_L and k_R are related proportionality constants to determine the level of the element $M(S)_n$ within the encoded left and right channels.

1) Output implementation: For purposes of this implementation, g is a scalar value applied per pair (or to the center microphone). The proportionality constants k for the flanks follow a standard sine-cosine panning function. The center microphone uses $k = 1$.

The g scalar is exposed to the user as a value α, where α is a number in decibels and $\alpha = [-20, 0]$.

Converting α_m to a scalar g_m can be done:

$$g_m = 10^{\frac{\alpha}{20}} \tag{26}$$

For the apportionment of the signal to the mixdown channels, the user inputs a value, β, where $\beta = [0, 1]$. The

value for β is used to indicate the separation of the virtual microphone pair into their corresponding mixdown channels. Lower values of β indicate that both microphones in the pair should increasingly come out of both mixdown channels; whereas higher values indicate more separation into their corresponding channel. The relationship with k can be shown as:

$$\begin{bmatrix} k_1 \\ k_2 \end{bmatrix} = \begin{bmatrix} \sin k(\beta) \\ \cos k(\beta) \end{bmatrix} \mid k(\beta) = \beta\frac{pi}{4} + \frac{pi}{4} \tag{27}$$

Following this, for any virtual microphone pair in M, the placement of the encoded sounds within the output mixdown follows:

$$\begin{bmatrix} y(L) \\ y(R) \end{bmatrix} = \begin{bmatrix} \sin k(\beta) & \cos k(\beta) \\ \cos k(\beta) & \sin k(\beta) \end{bmatrix} \cdot g \begin{bmatrix} m(S)_L \\ m(S)_R \end{bmatrix} \tag{28}$$

While $m(S)_{center}$ is simply:

$$\begin{bmatrix} y(L) \\ y(R) \end{bmatrix} = gm(S)_c J_{2,1} \tag{29}$$

Thus, the final, encoded output of the processor can be represented as:

$$y(M,S) = \sum_{i=1}^{|M|} y(M(S)_i) \tag{30}$$

Which, when taken with the relationship to the UI in (27) and (28), (30) can be expanded to:

$$y(M,S) = \begin{bmatrix} y(L)_{\text{mains}} \\ y(R)_{\text{mains}} \end{bmatrix} + \begin{bmatrix} y(L)_{\text{flanks}} \\ y(R)_{\text{flanks}} \end{bmatrix} + \begin{bmatrix} y(L)_{\text{center}} \\ y(R)_{\text{center}} \end{bmatrix} \tag{31}$$

V. Δt Compensation

The program needs to be able to provide time delay compensation for a sound source, as well as be able to only provide intermicrophone time of arrival cues. The method previously outlined for defining Δt will compensate for the "time of flight" delay based on the speed of sound to each microphone. However, a case needs to be defined for when this is not desired behaviour, and no time adjustment is desired.

Setting $\Delta t = 0$ would mean the removal of all time-domain cues from the encoding, instead the smallest distance for any microphone to the sound source, $d_{\vec{j}}$ where $j \in M \mid d_{\vec{j}} \leq d_{\vec{i}}, \forall i \in M$, can be subtracted from $d_{\vec{v}}$.

Using a user-definable boolean parameter, δ, the solution for Δt can be expanded into two cases:

$$\Delta t = \begin{cases} \frac{d_{\vec{v}}}{343+\Delta c} & \text{if } \delta = 0 \\ \frac{d_{\vec{v}} - d_{\vec{j}}}{343+\Delta c} & \text{if } \delta = 1 \end{cases} \tag{32}$$

VI. Approaches to the Abstraction of the Damping of Sound Pressure

References

[1] R. King, *Recording Orchestra and Other Classical Music Ensembles*. New York City, NY: Routledge, 2017.

[2] M. Williams, "The stereophonic zoom," tech. rep., Microphone Data, 2010.

[3] M. Williams and G. L. Du, "Multichannel microphone array design," tech. rep., Microphone Data, 2010.

[4] D. Arteaga, "Introduction to ambisonics," tech. rep., Dolby Laboratories, Inc., 06 2015.

[5] J. Meyer, *Acoustics and the Performance of Music*. Modern Acoustics and Signal Processing, Springer, 5th ed., 2009.