ADAPTING AUDIO MIXING PRINCIPLES AND TOOLS TO PARAMETER MAPPING SONIFICATION DESIGN

Prithvi Ravi Kantan, Sofia Dahl

Department of Architecture, Design and Media Technology
Aalborg University
Copenhagen, Denmark
{prka,sof}@create.aau.dk

Erika G. Spaich

Department of Health Science and Technology
Aalborg University
Aalborg, Denmark
espaich@hst.aau.dk

ABSTRACT

Designing a parameter mapping sonification (PMSon) involves defining a mapping function that determines how data variables affect audio signal parameters. The mapping function is represented using mathematical notation and/or characterized in terms of scaling, transfer function and polarity; both approaches manifest in software platforms for PMSon design. Math notation is not always directly relatable to complex design requirements, and simple characterizations lack generality and may be ambiguous - both issues hamper mapping function design, conceptualization, and dissemination. We seek to address them through knowledge transfer from audio mixing, a mature craft with strong parallels to PMSon design. For mixing, it was a versatile and universally applicable technological platform (the multitrack mixer) that supported the development of mixing technique, concepts, and recent formalizations thereof, laying the foundation for modern audio production. We posit that a PMSon design platform that adapts the essential elements of the mixer can similarly reinforce PMSon by supporting a mapping function representation directly tied to the design process. We define the correspondence between mixing and PMSon design, outline specifics of mixer functionality adaptation, and demonstrate the resulting capabilities with our proof-of-principle platform *Mix-N-Map* that is currently pending user testing. We believe a general PMSon framework explicitly rooted in audio mixing can potentially advance theory and practice to the benefit of PMSon designers and users alike.

1. INTRODUCTION

Sonification is the technique of converting data relations to perceived relations in an acoustic signal for the purpose of communication and interpretation [1]. Useful information has been deduced from acoustic patterns for centuries, for instance using the stethoscope as a diagnostic tool [2] or the Geiger counter for measuring radiation levels [3]. Here, the information-sound relationship is governed by the laws of physics and sound propagation. Powerful computers later made it possible for digital information to be converted to sound in almost any conceivable manner, laying the foundation for the proliferation of the sonification field [4, Chapter 1]. Almost 30 years after its formal inception, there remains little collective doubt within the community as to the potential of sonification and auditory displays to benefit humans in varied and significant ways [4, 5]. Sonification has been successfully explored in a myriad of domains ranging from space physics [6] to physiotherapy [7] and from geology to medical diagnosis[8]. The harsh flipside is that sonification has yet to enjoy mainstream adoption, with a number of likely reasons spanning the lack of consistent design guidelines [9, 10, 11], scientific evaluation standards [12, 13], and general-purpose technological platforms for researchers and designers alike [10, 14, 15, 16].

There are various sonification techniques, the most common being *parameter mapping sonification (PMSon)* [5, 17]. PMSon involves displaying multidimensional datasets through causal associations *(mappings)* between data variables and auditory perceptual parameters (e.g. pitch, loudness). PMSon has diverse applications ranging from data mining to assistive technology and music composition [17]. Designing a PMSon essentially concerns defining the algorithm that connects the data domain and the perceptual domain, or in other words, determining how data features and sound characteristics should be linked so as to satisfy a set of application-specific technical and aesthetic requirements [17]. We refer to the practitioner defining the algorithm as the sonification designer. As there are virtually infinite ways to map data to audio, PMSon provides enormous opportunities to create an appropriate sonification for a given purpose, but this freedom also creates challenges in terms of display consistency and comprehensibility [17].

1.1. Mapping Function Representation v/s Design

By definition, a PMSon requires a *mapping function (MF)* that appropriately connects the data domain (comprising data variables) to the audio domain (comprising audio signal control parameters - *audio parameters* for short) [17, 18]. The MF may comprise multiple individual mappings and as in [19], we define a single mapping as a "function from a subspace of the data domain to a subspace of the auditory domain". Hermann [18, Chapter 3] formalized a generic PMSon as:

$$s(t) = \sum_{i=1}^{N} f(g(x_i), t),$$

where x is a prepared data array of length N, s is the generated sound signal, f is the sound-generating function, and g is the MF. If the data array is treated as a signal flowing through the PMSon, one may say the MF g generates *control signals* for f. In line with this ethos, recent sonification design toolkits and platforms

[15, 16, 20, 21, 22] allow the MF g as well as individual mappings to be defined as equations in code. Given the wide range of operations (resampling, filtering, compression, polarity inversion, scaling, context generation) that a MF may encapsulate, such notations certainly possess the necessary precision and generality [17]. *Does this mean that it is best for PMSon designers to directly work with math equations?* Firstly, such notations are seemingly seldom used to describe MFs in most PMSon literature. Grond and Berger have pointed out that the benefits of math notation typically come at the cost of a steep learning curve [17], and this is especially relevant in an interdisciplinary field like sonification [4, Chapter 7] where many designers likely have non-STEM backgrounds. Secondly, such notations may not be designer-intuitive either. Consider the task of designing an equalization curve during audio mixing, which involves defining a filter transfer function for an input signal. Doing this clearly does not require extensive knowledge of control theory or signal processing algebra as professional mix engineers, despite commonly lacking said knowledge, are highly adept at defining complex filters that amplify or attenuate spectral characteristics as desired [23, Chapter 11]. Formally speaking, the biquadratic filters used in many digital equalizers have a generic transfer function $H(z)$ in the z-domain as follows [24, Chapter 5]:

$$H(z) = \frac{a_0 + a_1 z^{-1} + a_2 z^{-2}}{1 + b_1 z^{-1} + b_2 z^{-2}}$$

where the a and b coefficients directly define the frequency response of the filter. Although filter design is ultimately about defining these coefficients, the link between the raw coefficient values and the user-relevant filter characteristics (i.e. cutoff frequency, q-factor) is not intuitive and often involves elaborate equations [24, Chapter 5]. Conventional equalizers usually allow direct control over the user-relevant characteristics, modifying a and b coefficients 'under-the-hood' accordingly [24, Chapter 5]. As MF design for PMSon involves comparable processing techniques, it is plausible that math notation may *not* always be readily relatable to concrete design goals. Another issue is that such notations can become cumbersome with even moderate increases in sonification dimensionality and topological complexity.

Short Form	Movement Feature	Mapped Perceptual Parameter	Param. Range	Mapping Polarity	Mapping Func Order	Param. Smoothing Cutoff Freq.
S	*CoM Speed*	Flute Blowing Pressure	0 - 1	+	0.45	6 Hz
P	*Distance from Stand*	Flute Melodic Pitch	A4 - A5 note	-	0.75	No smoothing
F	*Freezes during Rise*	Bell Sound	Off - On	+	1	No smoothing
J	*Shank Jerk*	Flute Pitch Multiplier	1 - 10	+	0.45	19 Hz

Figure 1: An example assignment table representing a multidimensional PMSon [25].

The *assignment table* has been suggested as a more readable MF representation [17, 18] that specifies all mapped variable combinations along with scaling bounds for the data and audio signal parameter values in a tabular format (example in Fig. 1). Mapping polarity and transfer function shape are important considerations during design [17] that directly affect PMSon behavior and may also be included in the table. Such representations have formed the basis of MF design interfaces in notable software platforms such as the *Sonification Sandbox* [26], *Highcharts Sonification Studio* [27], and the *WAXML Sonification Toolkit* [28], often with individual mappings configurable via tab-based GUIs. While more intuitive and directly relatable to MF design tasks, assignment tables are prone to ambiguity in their interpretation and lack the generality and versatility of math notation. For instance, if a table specifies that a data variable undergoes smoothing (a linear operation) and nonlinear transformation, the order of these operations (not trivial) is unclear and may vary for different mappings - something difficult to capture in a table having fixed column order. Correspondingly, existing GUI-based MF design interfaces [26, 27, 28] typically do not seem to specify these details or allow the order of processing operations to be flexibly altered.

1.2. Drawing Basic Parallels to Audio Mixing

Thus far, common MF representations have directly manifested in the design of sonification design platforms, directly impacting the design of sonifications themselves as well as their eventual dissemination and reproduction. Given this far-reaching impact of the MF representation, it is important for the latter to balance the precision, generality and versatility of math notation with the accessibility and intuitiveness of assignment tables. It is in this precise context that we believe that there are valuable lessons to be learned from the mature field of audio production, specifically multitrack audio mixing. *Mixing* is the process by which multitrack audio material (e.g. music or film sound) is balanced, treated, and combined into a multichannel format [29, Chapter 1]. Mixing in the digital domain can, in a sense, be considered as the creative process of defining an algorithm that converts a raw multitrack audio 'dataset' into a final polished audio product. The mix engineer must creatively devise a set of audio processing operations to be applied to the raw multitrack before it is routed or fed to a playback device. The process is constrained by a set of technical / creative requirements and guided by critical listening [30]. The sum total of applied processing operations equates to a single complex transfer function (i.e. the mix) linking the multitrack to the playback device (mono / stereo / multichannel). Drawing parallels to PMSon (audio multitrack $\rightarrow$ non-audio dataset, playback device inputs $\rightarrow$ digital audio parameter controls), the correlate of the mix is the MF of the PMSon. In other words, the MF output in a PMSon 'drives' audio sources similarly to how a mixed multitrack drives playback devices. Given that the generic and now ubiquitous formula of an audio mix is readily applicable to various types of applications (film / stage sound, music, game sound) and genres within each, can such a formula be adapted for designing and representing MFs in the broad context of PMSon?

1.3. PMSon Mapping Function Design v/s Audio Mixing

We clarify certain key general distinctions between the processes. Most importantly, mixing is a highly subjective and artistic process typically focused on creating a pleasing and cohesive sonic output [23, 29], while PMSon MF design focuses on relaying the informational content of complex data to the audio signal domain such that it is effectively conveyed to the listener through temporal variations in sound characteristics [17]. Although both processes involve a combination of technical and aesthetic considerations [30, 31], these considerations may hence be prioritized very differently. Separate audio tracks during mixing usually also correspond to distinct sounds [23, Chapter 5], whereas several data variables in PMSon may be encoded onto different features of a single sounding entity [17]. These differences notwithstanding,

the two algorithms *(mix / MF)* exhibit the following fundamental similarities that, in our estimation, allow the meaningful adaptation of mixing principles and tools to MF design:

1. *Input to Algorithm:* The input to both algorithms is a dataset *(audio / non-audio)* (1) containing a finite number of numeric arrays *(audio tracks / data arrays)*.
2. *Output of Algorithm:* The output of both algorithms is one or more signals *(processed audio signals / audio parameter control signals)* that excite a subsequent signal processor *(playback device / sound source)* so as to optimally transmit the informational content of the input data through sound. The number of independent output signals may be lower or higher than the number of input arrays.
3. *Data Processing Methodology:* Each input array *(audio track / data array)* must undergo varying amounts of signal processing (linear or nonlinear, time-invariant or varying) in isolation and combination.
4. *Design Methodology:* The algorithm is defined by a human practitioner *(mix engineer / sonification designer)* on a case-by-case basis depending on the data type and content as well as the needs of the target user group. The design process involves solving a requirement-specific optimization problem and involves an iterative process of algorithm adjustment and critical listening.

1.4. Lessons from Audio Mixing

While MF representations have tended to inform the design of technological tools for PMSon design, the causality was the exact opposite for audio mixing in that *it was the technological tool (mixer) that indirectly informed the conceptual representation of the mix*. The art of mixing has existed since the adoption of the first multitrack tape machine and analog mixer in the 1960s [29, Chapter 1], but the first mathematical formalization of the mix algorithm was interestingly only established in 2014 by Terrell *et al.* [32], by which time mix engineering was already an established profession with well-understood processes and techniques that, incidentally, formed the basis of Terrell *et al.*'s formalization. The latter has primarily been cited in relation with machine learning-based mix automation and not conventional mixing practice.

It is noteworthy that mixing tools and techniques were able to mature without a formal representation of the mix algorithm, and even more so that complex and perfectly reproducible mixes have been achievable since the early 1970s [33]. This has been possible because the mix has historically been captured and represented by snapshots of the *configuration state of the audio mixer*, the tool universally applied during mixing [34, Chapter 11]; in other words, the *tool* has served as a scaffold for the representation of the mix. The state of the mixer usually comprises a highly complex set of signal processing parameter values, gains, signal routing specifics, and signal combination weights. Representing a mix in these terms combines the advantages of mathematical notation (reproducibility and generality) with the readability of assignment tables; for example, it is much easier to judge how a chain of filters affects an audio signal from their technical specifications (e.g. cutoff frequency, quality factor) depicted in a signal flow diagram than through direct inspection of their cumulative transfer function in math notation. For PMSon, it may be that the optimal MF representation similarly hinges on the inception of the ultimate design tool, and the mixer (the analog console and its software equivalent) may provide vital inspiration on this front.

On the usefulness / ease-of-use grid [35], the archetypal mixer undoubtedly falls in the region of 'super tools' as it allows its user to efficiently access vast regions of the mix design space while maintaining a relatively high level of usability [34, Chapter 11]. Since its first ever incarnation in 1958 [33], the mixer has co-evolved with mixing practice and its core formula (seen across hundreds of its variants) has endured well into the recent digital age, which is no mean feat given the recent rate of technological innovation. Our appraisal is that the power of this formula lies in the following specific aspects of its layout and functionality:

1. *Layout:* The user interface of a mixer directly mirrors the underlying signal flow architecture with independent audio signals assigned to visibly independent parallel channels and the respective signal processing controls arranged in order and easily accessible, with visual monitoring available at important stages of processing [34, Chapter 11]. The invoked *flow* metaphor is intuitive and greatly accelerates targeted user navigation of complex projects.
2. *Functionality:* Audio signals can be processed and combined in virtually any conceivable manner. Most mixers boast an array of inbuilt signal processing tools to flexibly manipulate the spectra, dynamics, phase, levels, and spatial characteristics of signals in isolation and combination, and mix-related design requirements can be achieved by chaining these processors together in certain ways [23, Parts 3,4]. Given the generic nature of these processors, a wide range of creative and technical tasks can be carried out using combinations of these processing *building blocks*. The digital domain has further expanded the flexibility of the formula by allowing virtually unlimited audio channels with a myriad of inbuilt and third-party processors that can be applied in any order to ergonomically realize design requirements [34, Chapter 6]. Perhaps most importantly, the processors operate in real time and the result of manipulating a processor is immediately audible, allowing efficient iterative modification [34, Chapter 6].

The core idea of adapting audio production tools for sonification design is not altogether novel, especially given how existing platforms have refashioned certain key elements of digital audio workstations for data sonification purposes [20, 22, 28]. These platforms have primarily adapted the data view and playback control elements rather than the mixer architecture and functionality. It may also be said that most PMSon platforms developed since the 1990s implicitly take inspiration from the mixer in that data streams serve as channel inputs and are processed prior to mapping, but not one recreates the signal processing versatility of a modern mixer. Furthermore, we believe that there is good reason for the mixer adaptation to be explicit and faithful as many PMSon designers have some background in music and music production [9] and may be able to effectively transfer these skills to MF design. We argue that just as mixing practice matured following the advent and evolution of the mixer, PMSon may benefit commensurately from a 'mixer' equivalent in terms of design practice, theory, and eventual output.

The remainder and primary contribution of this work is a detailed account of how the core elements of a mixer can be refashioned into a generic tool for MF design. A preliminary demonstration of the working and capabilities of such a tool (our own *Mix-N-Map* platform) is also provided.

2. ADAPTING KEY AUDIO MIXER COMPONENTS

We proceed to outline the relevant core components of the archetypal mixer, how they contribute to the mixing workflow, how they can benefit PMSon design workflows, and finally how they can be reused / adapted for generalized MF design.

Real-Time Operation and Data Input: In general, the mixer is a real-time device that spontaneously generates an array of output audio signals from an array of input signals as per its configuration. To maximize interoperability between different audio source types and allow the mixer to function in a source-agnostic manner, the signal inputs of a hardware mixer can work with various signal voltages (e.g. mic / line level), and various combinations thereof can be voltage-matched through preamplification [34, Chapter 11]. In software, audio signals are normalized between -1 and 1 regardless of their original sources [24, Chapter 1]. Compatible audio sources may be any combination of pre-recorded audio signals (from analog tape / audio files) and live-performed sound captured by microphones or instrument pickups, preamplified, and digitized [34, Chapter 11]; all sources are equal from the mixer's perspective. The *standardization of input voltages* and *source-agnostic real-time operation* of a mixer allow almost any conceivable combination of sound sources to be realized, which facilitates a diversity of real-time interactions between multiple performers and pre-recorded sounds. This makes mixers useful in live concert settings as well as for studio recording and overdubbing in various contexts [34, Chapter 11].

Along these lines, a PMSon design platform capable of processing and mapping any combination of real-time and pre-recorded data sources would readily lend itself to a range of interactive and non-interactive applications [4, Chapters 11,16]. Certain adaptations are necessary; a standardized real-time data input protocol (e.g. Open Sound Control (OSC) for real-time input, CSV or DAT for files) must be enforced - a potential challenge when interfacing non-standardized input devices. Moreover, the design platform must ensure jitter-free synchronization between real-time and pre-existing data sources, which requires a stable internal clock with a sufficiently high sampling rate to capture the relevant informational content within the data - something that may vary between sonification use-cases. All data inputs must also be normalized to a common value range to enable the meaningful combination of variables with distinct ranges.

Signal Flow Architecture: In a mixer, individual audio signals can be assigned to separate track channels for processing in isolation, the outputs of which can be *routed* or funnelled to a typically smaller number of *group* channels where specific additive combinations of the processed inputs can be further processed. In addition, mixers usually have a set of *auxiliary* channels that are typically used to duplicate input signals for parallel processing (e.g. with spatial effects). Track, group, and auxiliary channel output signals can all be routed to the master output, which combines them based on an assignment matrix to yield a mono (1 channel), stereo (2 channels) or multichannel audio signal that is played back through loudspeakers [34, Chapter 11]. Real-time signal processing operations can be applied at every stage of signal routing (track, group, auxiliary, master), and their configuration can be modified interactively while audio flows through them with immediate effect on the output sound. In most digital mixers, the processing operations can be applied in any order. The ability to process, combine, and route signals to the output as desired yields a massive degree of mix design flexibility, making mixers useful when working with musical ensembles of various sizes across genres and output media (stereo, surround, etc.) [34, Chapter 11]. Moreover, the ability to instantly audition the result of signal processing adjustments creates powerful sensorimotor control loops that expedite the mixing process as well as tool use learning in general [36].

The mixer signal flow architecture is readily transferable to a PMSon design platform. Normalized input data signals should be assignable to discrete individual channels for processing in isolation. It should then be possible for processed data signals to be combined or duplicated in any conceivable manner as per the desired mapping topology (potentially through a matrix-based interface inspired by *SonART* [37]) and further processed. All signal processing operations should be modifiable in real-time, and the final array of processed signals at the *output* of the MF should then be assignable to audio signal parameter controls (the correlate of playback device inputs for a mixer). In terms of necessary adaptations, there may be more outputs (mapped audio parameters) than input data sources (e.g. in a one-to-many mapping) and each parameter may have different value ranges as well as parameter-specific settings. A plausible architecture and corresponding interface is shown in Fig. 2.

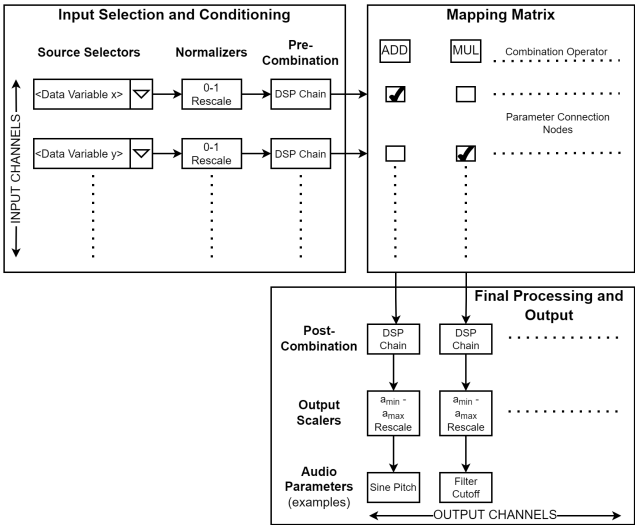

Figure 2: Data input and parameter control output sections linked by a mapping matrix. $a_{min} - a_{max}$ is the scaled range of interest of each mapped audio parameter.

Signal Processing Architecture: Although quite variable among mixers, most usually provide a set of inbuilt signal processing units that manipulate specific signal characteristics (amplitude, phase, spectral shape, dynamics, etc.) on a track / group / master channel [34, Chapter 11]. Each unit usually has a set of hyperparameters that can be adjusted in real time to tailor the processing to the needs of the signal and situation. In old analog mixers, the processing units were arranged in a fixed order and were quite limited [33], although these limitation has since been overcome in the digital domain [34, Chapter 6]. The essential suite of processing units includes gain controls (for controlling signal amplitude), polarity switches, equalizers (for spectral shaping), compressors (for time-domain dynamics shaping), saturation/distortion (for nonlin-

ear shaping), reverberation units, and delays (for spatial illusions) [23, Chapters 8-17]. Other common processors include effects such as chorus, phaser, vibrato, tremolo, and ring modulation [23, Chapter 18]. Situation-specific requirements are met by cascading multiple processing units, and this *building block*-based paradigm allows skilled engineers to achieve any desired signal manipulation.

Many of the aforementioned processor types can readily be refashioned as MF components in PMSon design. For instance, polarity switches can be used to manipulate mapping polarity, equalizers for data smoothing / noise reduction, and saturation / nonlinear distortion for applying nonlinear transfer functions. In addition, envelope detectors (a key component of compressors [24, Chapter 13]) can be used to manipulate the rising and falling edges of data signals. A key consideration is that the purpose of data processing prior to mapping in PMSon is ideally not to transform the data into something different but to suitably *present* it to the mapped parameter control for optimal information throughput. Adaptations to the processors may include modifications to work at lower sampling rates as well as some degree of simplification. Finally, it should be possible to place the processors in any order on isolated or combined data signals prior to mapping.

Visual Inspection Possibilities: Mixers provide visual metering of audio signals at several stages of the signal flow architecture such as the input or output of track / group / master channels as well as individual signal processors on each. Channel meters are usually one-dimensional displays of short-term peak or RMS values of a signal [34, Chaper 11], although many signal processor interfaces generate task-specific plots such as pitch contours, oscilloscope displays, short-term spectra, etc. [23, Chapter 3]. Although mixing is primarily guided by critical listening, the visual sense can provide valuable information to aid gain staging, attenuating unwanted resonances, configuring pitch correction, and troubleshooting signal flow issues, which can significantly expedite workflows [23, Chapter 3].

With PMSon, the possibility to inspect individual, combined, and processed data signals can similarly help the designer understand whether the applied signal processing is having the desired effect in certain technical tasks (e.g. smoothing noisy data or applying nonlinear transformations). This can aid decision making and troubleshooting, although the bulk of MF-related decisions should ideally be guided by listening. It ought to suffice for the platform to provide meters showing the overall raw inputs and output parameter control signals along with the signals entering and leaving each individual signal processor. The optimal mode of presentation will likely vary on a case-by-case basis depending on the type of data and how it encodes the information of interest. Given typical monitor framerates, data variables that vary with no important frequency components above 30 Hz at the chosen playback rate can be represented simply using 1-D meters indicating their instantaneous value, while spectral displays may be more informative for signals with higher frequency components of interest. Short-term oscilloscope-type plots may give a clearer idea of time-domain signal shape and can also be considered.

Output Signals: When it comes to transmitting mixed audio signals to sound reproduction devices (e.g., speakers), mixers usually allow a range of channel configurations from mono to multi-channel (e.g. surround), and the analog outputs of a mixer (audio interface for digital setups) are standardized at line-level, making them compatible with a wide range of playback devices (e.g. consumer stereo speakers, public address systems, headphones, professional speakers, television speakers, etc.) [34, Chapter 11]. This *destination-agnostic* nature of the outputs and physical dissociation from the playback device make mixers suitable for a variety of applications and output media.

The MP output signals serve as control signals that *drive* the 'playback devices' - digital audio sources. A *destination-agnostic* MF design platform whose output signals are compatible with a wide range of digital audio sources enables virtually infinite PMSon design possibilities. This can be realized if the platform enables external transmission of control signals (via a standard protocol such as OSC) to a digital audio workstation where they can be mapped to any combination of control parameters spanning those of third party synthesizers, audio effects and audio playback. For example, REAPER[1] readily receives OSC messages and can map separate OSC streams to selected session parameters using the ReaLearn[2] extension.

Representation, Storage, Recall: Most digital and digitally-controlled analog mixers make it possible to store a *snapshot* of their state to digital memory and recall it in full at a later stage [34, Chapter 11]. The mix snapshot hence serves as a complete and reproducible representation of the mix algorithm that is directly tied to the design process (mixing). The algorithm can be disseminated as a 'preset' that can be loaded and run in compatible software and/or as a detailed signal flow schematic that outlines all signal processing and combination operations as-is. Such a representation, when applied to a PMSon MF, may address the aforementioned issues with math notation and assignment tables.

3. PROOF-OF-PRINCIPLE: MIX-N-MAP

Mix-N-Map is our mixer adaptation for MF design built using the JUCE[3] framework. It was iteratively developed over a 3-year period with the purpose of facilitating movement sonification design for motor rehabilitation. Past versions of *Mix-N-Map* have demonstrated the versatility of its general formula in several of our studies on sonifying complex movements like walking and sit-to-stand [25, 38]. The MF design interface is shown in Fig. 3, and the labelled functional elements in the figure are referred to in connection with the core components covered in the previous section (demo video link at end of section).

Data Input: Mix-N-Map allows data input from CSV-formatted files, real-time sources (a set of supported wearable motion sensors or OSC streamed over UDP), or any combination thereof, configurable via a separate interface (not shown but part of platform). Once set up, the available data variables can be 'played back' in a loop at a user-defined rate within the desired time bounds (section *6* in Fig. 3), or in the case of real-time sources, streamed at a maximum sampling rate of 100 Hz.

Signal Flow Architecture: Individual variables can be fed into the MF by assigning them to discrete input channels *(1a)*, where their values are metered in real-time with a horizontal 1-D meter (black background, white fill) *(1b)*, and their value ranges of interest can be precisely defined using a two-value slider *(1c)* and normalized to a standard 0-1 range. The normalized signal on each input channel is then fed to a DSP chain comprising five customizable processors *(1d)* placed in series, the output of which is fed to the input of the respective row of the mapping matrix *(2)*. The

[1] `https://www.reaper.fm/`
[2] `https://www.helgoboss.org/projects/realearn/`
[3] `https://juce.com/`

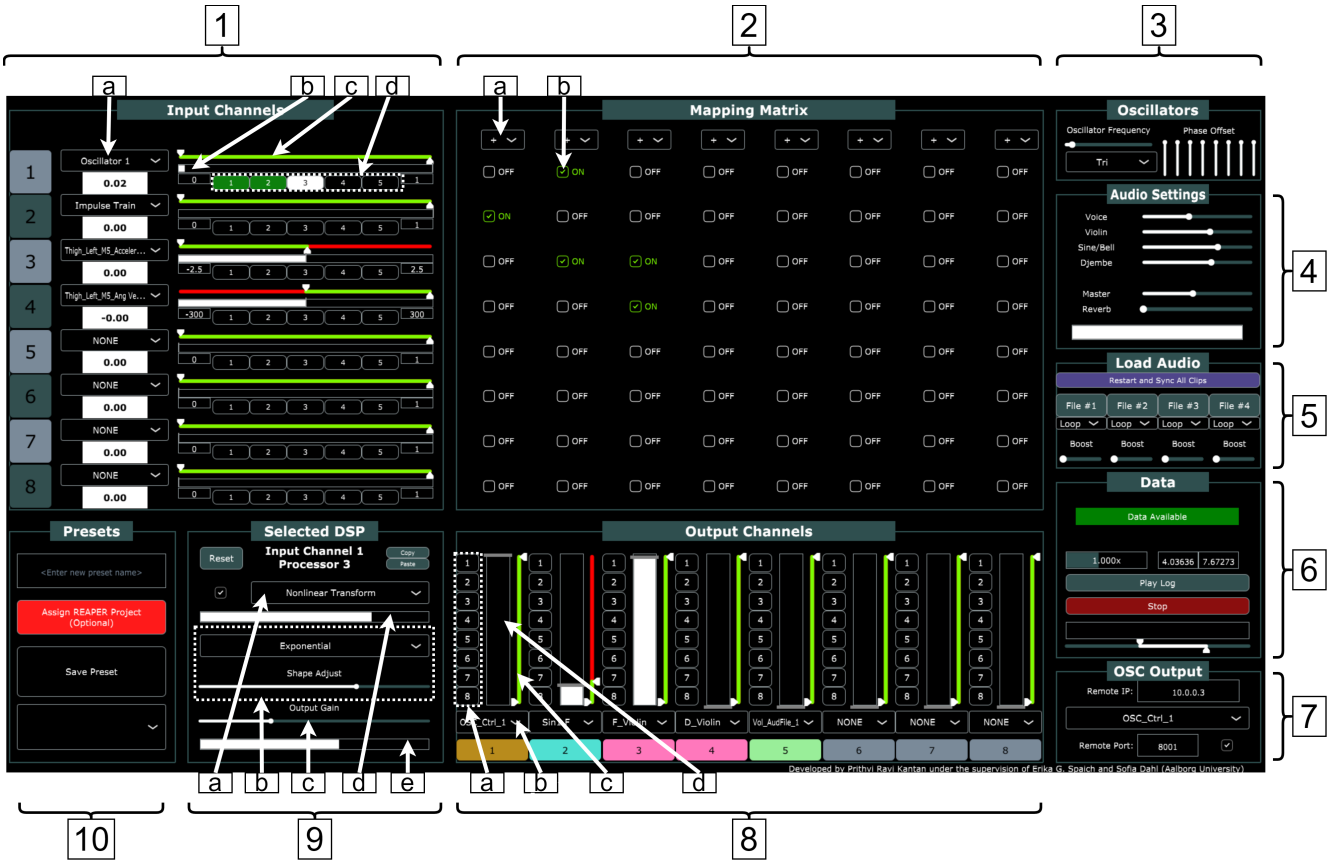

Figure 3: The *Mix-N-Map* interface with key functional elements labelled. **1** = data input channel section (a = source variable selector, b = DSP effect chain, c = input level meter, d = normalization bound setter), **2** = parameter mapping matrix (a = column combination operator, b = matrix node), **3** = low frequency oscillator bank, **4** = audio settings (mainly volume of inbuilt synths), **5** = external audio file handling, **6** = dataset playback controls, **7** = OSC control signal configuration, **8** = audio parameter output channel section (a = parameter selector, b = DSP effect chain, c = output level meter, d = scaling bound setter), **9** = selected DSP context menu (a = DSP operation selector, b = DSP operation-specific hyperparameter controls, c = post-DSP gain control, d = DSP input level meter, e = DSP output level meter), **10** = mapping preset handling.

parallel signals entering the mapping matrix are combined *vertically* based on the combination operator (summation or multiplication) defined for each column *(2a)* and the state of the matrix node checkboxes in each column *(2b)*, yielding a parallel set of signals that serve as the input to the audio parameter output channel section *(8)*. Each signal is fed to a DSP chain composed of eight customizable processors *(8a)*. Each output channel can be assigned to a destination audio parameter *(8b)*, and the output of the DSP chain can be scaled to the desired value range of the parameter using a two-value slider *(8c)* with the final output metered in real time *(8d)*. All variable / parameter assignments as well as the normalization and scaling bounds are adjustable on the fly. The current version has eight input and output channels, but this can be expanded through scrollable input and output channel sections.

Signal Processing Architecture: The DSP chains *(1b / 8b)* on each input / output channel allow multiple signal processing operations to flexibly be applied to the data in isolation / combination. Each channel has 5 / 8 processor slots that are set to 'passthrough' mode by default and can be clicked on for modification via a DSP context menu *(9)*. Here, the desired processor type (available options - filter, envelope follower, polarity inversion, nonlinear transformation, step quantizer, etc. inspired by mixer processors) can be selected from a list *(9a)* and processor-specific hyperparameters (e.g. nonlinear transform type or shape) can be adjusted *(9b)*. Every processor has an output gain slider *(9c)* inspired by the *make-up gain* control in many mixer processors, and all operations can be adjusted in real-time. The input and output signals of the selected processor can be monitored as well *(9d,9e)*, and signal saturation is clearly indicated in red.

Audio Generation and Interfacing: Each scaled output signal can be mapped to a range of audio parameter destinations accessible from a list *(8a)*. There are a set of real-time inbuilt synthesizers created using FAUST[4] DSP (e.g. sine wave, voice, bell, violin physical models) whose parameters (excitation amplitude, frequency, etc.) are directly selectable. Next, it is possible to load up to four external audio files *(5)* and map to their amplitude / triggering parameters. Finally, it is possible to send the signals

[4] https://faust.grame.fr/

to third-party software in the form of OSC messages by defining a remote IP address and UDP port *(7)*. This allows the interface to control virtually any digital audio source that accepts OSC input (e.g. REAPER via ReaLearn), opening up a massive range of audio synthesizers and parameter combinations.

Finally, the state of the interface can be stored as a mapping preset and recalled for future use *(10)*. In all, the compact interface and real-time operation of *Mix-N-Map* aims to enable a mapping design workflow similar to that afforded by audio mixers, wherein many signals can be inspected, manipulated, and combined to satisfy design requirements based on iterative adjustments and critical listening. The core functionality of *Mix-N-Map* is showcased in the provided **demo video**[5].

4. GENERAL DISCUSSION

In this work, we showed how a generic functional platform for mapping design in parameter mapping sonification can be built by adapting the core architectural elements of an archetypal audio mixer. We proceeded to demonstrate the functional capabilities of such a platform as well as the design workflow it affords through our own proof-of-principle platform. Our overall appraisal at the current time is that the biggest advantage of the 'mixer state' representation of the MF is that it maintains much of the generality, precision, and flexibility of mathematical notation [17] while simultaneously bridging the conceptual representation of the MF and its practical design process. Similarly to mixing, the design tool structure can hence serve as the scaffold for theory and design practice alike rather than the converse, paving the way (as it did with mixing) for the emergence of heuristically informed design theory to guide future generations of designers - something that the sonification field sorely needs [1, 17]. A example of such a guideline could be regarding what sequence of data DSP operations should be applied to noisy time-series data prior to pitch mapping (e.g., smoothing $\rightarrow$ compensatory gain $\rightarrow$ exponential transfer function $\rightarrow$ step quantization $\rightarrow$ scaling) and methods for optimally tailoring these operations to the data in real time. Similar guidelines for mixing and processing various sound source types have emerged over the years [23] (e.g., how to process a bass drum signal so that it is sufficiently punchy, snappy, and thick-sounding for a pop mix), and there is no clear reason why this cannot happen for PMSon if a suitable ecosystem for MF design is created.

A major strength of the explicit mixer analogy is the potential of flexible data DSP to transform raw data as per the requirements of a wide range of applications. It is well known that raw data often requires considerable and complex manipulation (data preparation and signal conditioning [17]) in order for its informational content to suitably be presented to an end-user through sound changes. We believe that the mixer-inspired palette of customizable data DSP building blocks applicable in any order makes it possible to condition data to suit the perceptual affordances of virtually any mapped audio-specific parameter to meet application requirements, just as modern mixers can cope with the requirements of various genres and audio production contexts [34, Chapter 11]. The flipside of this flexibility is that just like with mixing [23], such a workflow will most likely take years to master, especially in the current absence of concrete practical guidance. A clear limitation of the current work is that we have yet to evaluate *Mix-N-Map* with real-life users in a broad variety of PMSon application contexts, so it remains unclear how readily the mixer formula can cope with the existing range of real-world design requirements; this will be carried out as part of future work. The development and evaluation of a fully standalone platform with data waveform visualization, playback control, and audio environment integration is also planned. Supporting and upgrading such a platform post-release is challenging in its own right, but we hope that the merits of the mixer adaptation can take flight even the platform itself plunges into the usual short-term research project chasm [14].

[5] https://doi.org/10.5281/zenodo.11115100

5. CONCLUSION

Despite the critical differences between audio mixing and parameter mapping sonification design (purpose, specific design goals, input data type), they are fundamentally iterative search processes involving alternating phases of signal processing adjustment and critical listening. We believe that this core similarity enables decades of technical expertise and technological development in the audio mixing field to be explicitly transferred to the sonification field in an attempt to address long-standing issues in the field - the absence of generic design platforms and, in turn, the lack of a common vision or conceptualization of the mapping function. We provided a detailed account of how mixing tools and principles can be adapted for sonification design along with a description of our software platform founded in said principles. Although further research is needed in order to ascertain the efficacy of such a tool in a variety of sonification contexts, we believe that the core foundation of our premise is grounded in a proven formula. Hopefully, this can help the auditory display community more effectively access and operationalize the potential of sonification in daily life.

6. ACKNOWLEDGMENT

The contribution of PRK and SD was partly funded by NordForsk's Nordic University Hub, Nordic Sound and Music Computing Network NordicSMC, project number 86892.

7. REFERENCES

[1] G. Kramer, B. Walker, T. Bonebright, P. Cook, J. H. Flowers, N. Miner, and J. Neuhoff, "Sonification Report: Status of the Field and Research Agenda," 2010.

[2] P. Bishop, "Evolution of the stethoscope," *Journal of the Royal Society of Medicine*, vol. 73, no. 6, pp. 448–456, 1980.

[3] G. F. Knoll, *Radiation detection and measurement*. John Wiley & Sons, 2010.

[4] T. Hermann, A. Hunt, and J. G. Neuhoff, *The Sonification Handbook*, T. Hermann, A. Hunt, and J. G. Neuhoff, Eds. Logos Verlag Berlin, 2011.

[5] S. Barrass and G. Kramer, "Using sonification," *Multimedia systems*, vol. 7, no. 1, pp. 23–31, 1999.

[6] R. M. Candey, A. M. Schertenleib, and W. Diaz Merced, "Xsonify Sonification Tool for Space Physics," in *Proceedings of the 12th International Conference on Auditory Display, London, UK, June 20-23*, 2006.

[7] J. Guerra, L. Smith, D. Vicinanza, B. Stubbs, N. Veronese, and G. Williams, "The use of sonification for physiotherapy in human movement tasks: A scoping review," *Science & Sports*, vol. 35, no. 3, pp. 119–129, 2020.

[8] E. Brazil and M. Fernström, "The sonification handbook," T. Hermann, A. Hunt, and J. G. Neuhoff, Eds. Logos Verlag Berlin, 2011, ch. Navigation of Data, pp. 509–524.

[9] J. G. Neuhoff, "Is Sonification Doomed to Fail?" in *Proceedings of the 25th International Conference on Auditory Display (ICAD 2019)*. Newcastle upon Tyne: Department of Computer and Information Sciences, Northumbria University, June 2019, pp. 327–330.

[10] M. A. Nees, "Auditory graphs are not the "killer app" of sonification, but they work," *Ergonomics in Design*, vol. 26, no. 4, pp. 25–28, 2018.

[11] S. Roddy and B. Bridges, "Mapping for meaning: the embodied sonification listening model and its implications for the mapping problem in sonic information design," *Journal on Multimodal User Interfaces*, vol. 14, no. 2, pp. 143–151, 2020.

[12] A. Supper, *The Oxford Handbook of Sound Studies*. Oxford Academic, 2011, ch. The Search for the "Killer Application": Drawing the Boundaries Around the Sonification of Scientific Data.

[13] N. Degara, F. Nagel, and T. Hermann, "Sonex: an evaluation exchange framework for reproducible sonification," in *Proceedings of ISon 2013, 4th Interactive Sonification Workshop, Fraunhofer IIS, Erlangen, Germany, December 10*, 2013.

[14] A. Andreopoulou and V. Goudarzi, "Sonification first: The role of ICAD in the Advancement of Sonification-related Research," in *The 26th International Conference on Auditory Display (ICAD 2021), June 25-28 2021, Virtual Conference*, 2021.

[15] D. Reinsch and T. Hermann, "Sonecules: A Python Sonification Architecture," in *Proceedings of the 28th International Conference on Auditory Display (ICAD 2023)*. Linköping University, Campus Norrköping, Sweden, 2023.

[16] M. Poret, J.-M. Celerier, M. Desainte-Catherine, and C. Semal, "Proof of concept of a generic toolkit for sonification: The Sonification Cell in Ossia Score," in *Proceedings of the 28th International Conference on Auditory Display (ICAD 2023)*. Linköping University, Campus Norrköping, Sweden, 2023.

[17] F. Grond and J. Berger, "The sonification handbook," T. Hermann, A. Hunt, and J. G. Neuhoff, Eds. Logos Verlag Berlin, 2011, ch. Parameter Mapping Sonification, pp. 363–399.

[18] T. Hermann, "Sonification for Exploratory Data Analysis," Ph.D. dissertation, 2002.

[19] G. Dubus and R. Bresin, "A Systematic Review of Mapping Strategies for the Sonification of Physical Quantities," *PloS one*, vol. 8, no. 12, p. e82491, 2013.

[20] T. Peng, H. Choi, and J. Berger, "Siren: Creative and extensible sonification on the web," in *Proceedings of the 28th International Conference on Auditory Display (ICAD 2023)*. Linköping University, Campus Norrköping, Sweden, 2023.

[21] D. Reinsch and T. Hermann, "Interacting with sonifications: The mesonic framework for interactive auditory data science," in *Proceedings of the 7th Interactive Sonification Workshop (ISon), Hanse Institute for Advanced Study, Delmenhorst, Germany*, 2022, p. 65.

[22] S. Phillips and A. Cabrera, "Sonification workstation," in *The 25th International Conference on Auditory Display (ICAD 2019) 23–27 June, Newcastle upon Tyne, UK*, 2019.

[23] M. Senior, *Mixing secrets for the small studio*. Routledge, 2018.

[24] W. Pirkle, *Designing Audio Effect Plug-Ins in C++: with digital audio signal processing theory*. Routledge, 2012.

[25] P. Kantan, E. G. Spaich, and S. Dahl, "An embodied sonification model for sit-to-stand transfers," *Frontiers in psychology*, vol. 13, p. 806861, 2022.

[26] B. N. Walker and J. T. Cothran, "Sonification sandbox: A graphical toolkit for auditory graphs," in *Proceedings of the 2003 International Conference on Auditory Display, Boston, MA, USA, 6-9 July*, 2003.

[27] S. J. Cantrell, B. N. Walker, and Ø. Moseng, "Highcharts sonification studio: an online, open-source, extensible, and accessible data sonification tool," in *Proceedings of the International Conference on Auditory Display. International Community For Auditory Display, Atlanta, GA, USA*, vol. 7, 2021.

[28] H. Lindetorp and K. Falkenberg, "Sonification for everyone everywhere evaluating the WebAudioXML Sonification Toolkit for Browsers," in *Proceedings of the 26th International Conference on Auditory Display (ICAD 2021), June 25-28 2021, Virtual Conference*, 2021.

[29] R. Izhaki, *Mixing Audio: Concepts, Practices, and Tools*. Routledge, Oxfordshire, 2017.

[30] M. N. Lefford, G. Bromham, and D. Moffat, "Mixing with intelligent mixing systems: Evolving practices and lessons from computer assisted design," in *Audio Engineering Society Convention 148, June 2-5, Online*, 2020.

[31] F. Grond and T. Hermann, "Aesthetic strategies in sonification," *AI & society*, vol. 27, pp. 213–222, 2012.

[32] M. Terrell, A. Simpson, and M. Sandler, "The mathematics of mixing," *Journal of the Audio Engineering Society*, vol. 62, no. 1/2, pp. 4–13, 2014.

[33] "The History Of The Studio Console, Part 1 — vintageking.com," https://vintageking.com/blog/2023/11/history-of-consoles-part1, [Accessed 28-03-2024].

[34] D. M. Huber and R. Runstein, *Modern Recording Techniques*. Routledge, 2013.

[35] M. Keil, P. M. Beranek, and B. R. Konsynski, "Usefulness and ease of use: field study evidence regarding task considerations," *Decision support systems*, vol. 13, no. 1, pp. 75–91, 1995.

[36] T. Hermann and A. Hunt, "The Importance of Interaction in Sonification," in *Proceedings of ICAD'04 - 10th Meeting of the International Conference on Auditory Display, Sydney, Australia, July 6-9*, 2004.

[37] O. Ben-Tal, J. Berger, B. Cook, M. Daniels, and G. Scavone, "SonART: The Sonification Application Research Toolbox," in *Proceedings of the 2002 International Conference on Auditory Display, Kyoto, Japan, July 2-5*, 2002.

[38] P. R. Kantan, S. Dahl, H. R. Jørgensen, C. Khadye, and E. G. Spaich, "Designing ecological auditory feedback on lower limb kinematics for hemiparetic gait training," *Sensors*, vol. 23, no. 8, p. 3964, 2023.