VAS Objects - A Toolbox for (dynamic) convolution-based Applications for Pure Data and Max/MSP

Hannes Barfuss[1], Thomas Resch[2]

[1] *Forschung & Entwicklung, Hochschule für Musik Basel FHNW, E-Mail: hannes.barfuss@fhnw.ch*
[2] *Forschung & Entwicklung, Hochschule für Musik Basel FHNW, E-Mail: thomas.resch@fhnw.ch*

Abstract

The Virtual Acoustic Spaces (VAS) objects are a set of externals for Max/MSP (Max) and Pure Data (Pd) for (dynamic) convolution-based applications. While numerous objects already exist for static convolution for Max as well as for Pd, no object library available to date works with both applications while capable of handling up-to-date file formats (such as sofa), providing dynamic convolution with filters exchangeable in real-time, and being published as open-source software. Besides partitioned, multi-threaded convolution with static filters, the objects facilitate crossfade-based swapping between current and target filter (for applications such as dynamic binaural synthesis) as well as a true polyphonic dynamic convolution in which the respective filters may decay to the end. As the filter length is not limited, binaural synthesis based on binaural room impulse responses (BRIRs) is also possible.

Introduction

The objects are implemented with the VAS Library [1], a cross-platform, open-source C library for the development of convolution-based real-time applications. The library itself is written in plain C and therefore should compile on all operating systems and hardware platforms. In the initial stages of development, the focus was on the realisation of personal projects. Meanwhile it is explicitly intended as a tool for the community for realising media installations, Extended Reality (XR) applications and listening tests. Initially, three objects were developed for Pd only: A simple dynamic binaural synthesis, a uniformly partitioned convolution for reverberation and an object for headphone equalisation. These objects were originally programmed and used for the H.E.I. Guide, an interactive soundwalk taking place in Basel, Switzerland [2]. The soundwalk is based on the RWA - Real World Audio [3] software, which itself uses libPd [4] as its audio backend. Over the years, the VAS library has been extended to feature numerous new classes, functions, and application examples: A Unity plugin for dynamic binaural synthesis [5] based on binaural room impulse responses (BRIRs) was used in several XR projects, for example in [6] [7]. An object for dynamic convolution was added for the project "Authenticity in Music Production" [8]. And due to the wide use and the comfortable user interface, most objects have now also been adapted for Max/MSP.

This paper starts with a brief discussion of related work in section 2 and goes on to discuss implementation details considered by the authors to be of particular interest (section 3). Section 4 explains the objects and their usage. Some performance measurements have been carried out and are reviewed in section 5. Finally, section 6 gives an outlook on further development perspectives, possible additions, and optimisations.

Related Work

Several libraries and objects for Pd and Max exist for the realisation of convolution-based applications. The Soundscape Renderer (SSR) [9] is a C++ software capable of rendering dynamic binaural synthesis and other spatial audio formats such as wave field synthesis and Ambisonics. Besides source code and a stand-alone version, the authors also provide objects for Pd. The HISS tools are a set of objects for convolution-based applications for Max [10]. The focus of these objects is on reverberation and measurement of impulse responses rather than spatial audio synthesis. The earplug~ object is a binaural renderer for Pd using the KEMAR HRTF measurements [11]. The recently released SOFAlizer for Pd [12] is based on earplug~ but supports the SOFA format and therefore the usage of different head related transfer functions (HRTFs). Another object for Pd, Space~ implements a generalised real time model for spatial processing [13].

Implementation Details

Reverberation

For reverberation, the library supports a single-threaded equal and a multithreaded non-equal partitioned approach. Although parallel processing with growing partition sizes is usually significantly faster, there are special cases, for instance that described in [1], where equal partitioning can be more efficient. In other cases, hardware or software constraints might not permit multithreading at all. Therefore, the VAS Objects make both variants available to the user (vas_reverb~ and vas_partreverb~). The partition scheme for the non-equal version is close to Gardner's version as described in [14] (figure 5). Instead of starting with a slightly longer direct form convolution in the beginning and then doubling segment sizes every two frames for the frequency domain convolution, however, the VAS partition scheme starts with eight small segments of the same size at the beginning, as shown below, in figure 1.

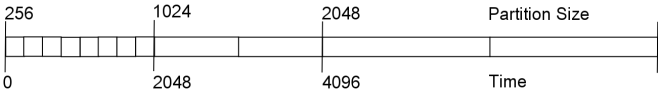

Figure 1: Partition scheme of the non-equal partitioned convolution.

Dynamic convolution

The necessary dynamic filter exchange for the dynamic convolution objects (vas_binaural~ and vas_dynconv~) is realised by calculating the convolution with both the current and the target IR simultaneously for at least one frame. An equal power crossfade is used for mixing the two signals.

Besides its application in binaural synthesis, dynamic convolution can be used for modelling analog hardware, such as guitar effect pedals or amplifiers. Static convolution with a user-settable impulse response has already been present in consumer products for some time, such as in the Helix guitar processor [15]. Since the impulse response cannot be exchanged on the fly, the possibilities remain quite limited and focus on emulating systems and devices whose impulse response does not change in real-time, such as guitar/bass cabinets [16] or the resonance chambers of acoustic guitars [17]. With vas_dynconv~, we present an easily configurable way to model (parts of) audio effect processor hardware with dynamic convolution. The following example illustrates one of the intended application scenarios for the vas_dynconv~ object.

Convolution can only reproduce the linear effects of a system. However, the vas_dynconv~ object can also be useful as a component in the emulation of non-linear systems. In a research project on preserving digital hardware [18], we emulated the Princeton PT2399 delay circuit [19], which is frequently used in effect floor pedals for guitar players. The PT2399 can be driven into instability and self-oscillation by increasing the gain of the signal fed back into the delay line. The resulting (highly non-linear) distortion as well as the potentially endless repetitions of the delay render a purely convolutional emulation of the PT2399 impossible. However, the sonic character of the non-linearities in the PT2399 is to a great extent due to its far-from-flat frequency response. The circuit contains a resonant filter with a variable cut-off frequency. By sampling the impulse response of this filter several times with different settings and crossfading between them with vas_dynconv~, we were able to approximate the sonic character of the PT2399. The delay was implemented with a simple ring-buffer and the vas_dynconv~ object was placed inside the delay feedback loop. In a series of informal blind listening tests, this hybrid emulation was in general considered “similar” or “very similar” to the original hardware. Since the PT2399’s filter is rather subtle, we used only 4 impulse responses, but the vas_dynconv~ object can handle IR sets of up to 360 IRs.

Usage

All convolution-based VAS-objects take the partition size as the first (optional) argument (default value is 256 samples). If it is set equal to the DSP block size, no additional latency is introduced. A larger number increases latency but decreases CPU load. Audio signals are always routed into the first inlet.

vas_binaural~

Impulse response (IR) sets (either in SOFA format or in a proprietary text format) can be passed as a second argument or loaded after object creation with the read message. The first argument of the read message is the IR set to be loaded. The other arguments are optional and are used to set the partition size, the offset for the IR set (in samples) and the end of the IR set (in samples). The length of the transform functions is not limited. Therefore, it is also possible to load longer BRIR sets. The two additional inlets expect azimuth and elevation which can alternatively set by messages. A usage example can be found in the help patch which is available both for Pd (as depicted in Figure 1) and Max.

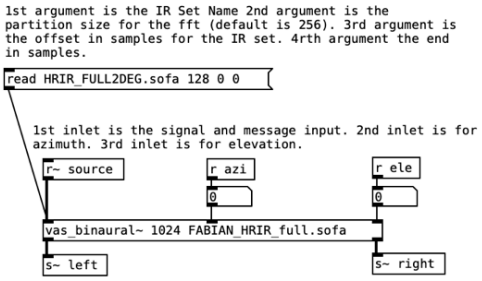

Figure 2: A Pd Example of the vas_binaural~ Object.

vas_reverb~ & vas_partconv~

The objects calculate a mono to stereo convolution. IRs can be passed as an optional argument or can be loaded with the read message as described above for the vas_binaural~ object. In addition, both objects support loading impulse responses with the set message from Pd arrays (in the case of Max from buffer~ objects) as shown in figure 2 below.

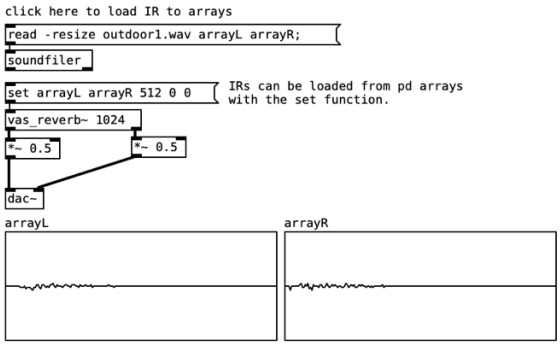

Figure 3: Loading IRs from Pd arrays.

vas_hpcomp~

As the headphone transfer function (HpTF) may strongly colorize a binaural signal [20], the VAS library provides a specific tool (vas_hpcomp~) for headphone compensation. The object performs a two-channel convolution. Compensation filters can be loaded as SOFA files, proprietary text files or from Pd arrays (Max buffers) as described above.

vas_dynconv~

The vas_dynconv~ object takes the partition size to be used for convolution as (optional) creation argument. IRs are expected to be loaded into *array* objects.

The *configure* message takes three arguments. The first one is the partition size, the second one is the maximum filter length (in samples) and the third one is the number of IRs to load.

It is possible to load an IR set that contains fewer IRs than are specified using the *configure* message. In this case, a *setandinterpolate* message should be sent, followed by the array names and their desired corresponding indices in pairs. In between those indices, IRs are calculated with linear interpolation. This can be useful for filter sets that would otherwise not be high-resolution enough to allow for smooth crossfading. By sending an *index* message followed by an integer, the vas_dynconv~ object executes a filter exchange and cross-fade from the current IR to the IR at the specified index.

Additional help and a simple hands-on example can be found in the help patch which is available both for Pd and Max.

Performance

vas_binaural~

Within the scope of the Immersive Audio Guiding System (IAGS) research project [21], a Unity Audio Plugin was derived from the vas_binaural~ object. The performance of this plugin has been tested against Resonance Audio Renderer [22] within the Unity game engine. For the test, a simple 3D scene was set up in Unity and deployed to various mobile devices. The number of rendered binaural sources was constantly increased to determine the maximum number of sources that could be rendered simultaneously without dropouts or other audio glitches. This test was executed both with Resonance Audio Renderer and the VAS Library. As all test devices have multi-core CPUs. The multi-threaded version of the VAS binaural engine was used for testing. In addition to the convolution engine, the VAS Library uses multiple filters per source to calculate material absorption and other room characteristics. Resonance Audio Renderer uses a signficantly different room acoustics model than VAS Library (for details, consult Resonance Audio's source code [23]). This should be considered when interpreting the performance test results in Figure 4.

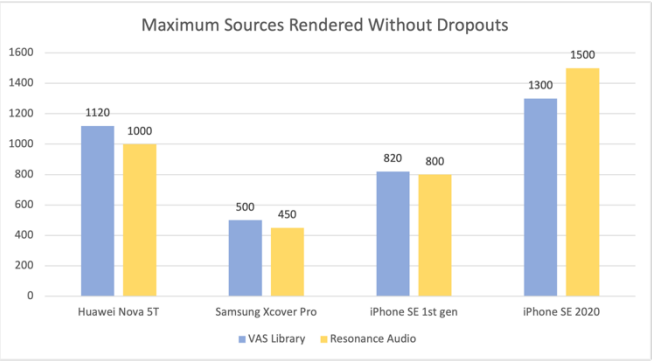

Figure 4: Performance Test Results of VAS Library and Resonance Audio Renderer.

The results show that on iOS, Resonance Audio Renderer can calculate more sources than the VAS Library. However, on Android, the VAS Library outperforms Resonance Audio. The results also show an overall gap between iOS devices and Android devices with comparable hardware, both with Resonance Audio and with VAS Library. The reason for the relatively poor performance of Android devices is not yet entirely clear. OS-dependent differences in Unity's audio engine implementation or between the low-level Audio APIs of the two operating systems (CoreAudio on iOS vs. AAudio or OpenSL on Android) could be responsible. Since fast fourier transforms (FFT) account for a relatively large part of the overall calculations, performance differences between the FFT algorithms used could also significantly affect the overall performance.

vas_dynconv~

In the case of vas_dynconv~, we used a Raspberry Pi 4 (Model B) to calculate dynamic convolution with 4 impulse responses of 512 samples length each, with a partition size of 32 for minimum latency (< 1ms). No glitches, dropouts or other artefacts were encountered. This shows that the vas_dynconv~ object is suitable for emulating effect processor foot pedals since it can satisfy both low latency and low space requirements. However, no formal performance tests have been carried out with the vas_dynconv~ object yet, and the described usage example is far from exhausting the CPU resources of the Raspberry Pi.

Conclusion and Outlook

Four Max and Pd externals for convolution-based applications have been presented. So far, room acoustics can only be modelled by using a set of BRIRs. However, the Unity Plugin developed for the IAGS research project already contains a room acoustics model that can calculate up to 1'000 binaural early reflections in real-time. The reflection parameters such as position, delay or material absorption filtering are automatically calculated based on the position of the listener and the surrounding walls. This room acoustics model will soon be added to VAS Library.

Further research is needed to examine the potential of dynamic convolution for the modelling of audio effect processors and/or the preservation of such devices once their original electric components become unavailable. There are several approaches for convolutional modelling of such devices. First, a purely convolutional approach can be chosen for linear systems. However, real-world audio effect processors are rarely linear systems. Second, Volterra kernels can be used to describe and simulate a weakly non-linear system [24, 25, 26]. Third, this paper presented a hybrid approach that splits a system into its linear and non-linear components. The linear components are then reproduced with dynamic convolution, while the non-linear components are approximated algorithmically. This approach could be refined in several ways. For example, the system's impulse response could be observed with different input signal levels.

References

[1] Resch, T., Böhm, C., Weinzierl, S.: VAS – A cross platform C-library for efficient dynamic binaural synthesis on mobile devices. AES, International Conference on Headphone Technology, San Francisco, (2019).

[2] Hauert, S.: H.E.I. Guide, URL: https://heiguide.ch/

[3] Resch, T., "RWA – A Game Engine for Real World Audio Games", in Proceedings of the International Conference on New Interfaces for Musical Expression, Baton Rouge, (2015).

[4] Brinkmann, P., Kirn, P., Lawler, R., McCormick, C., Roth, M., & Steiner, H. C.: Embedding pure data with libpd. In Proceedings of the Pure Data Convention, Vol. 291, (2011).

[5] Resch ,T., Hädrich, M. "The Virtual Acoustic Spaces Unity Spatializer with custom head tracker" in Proceedings of the 5th International Conference on Spatial Audio ICSA, Ilmenau, (2019).

[6] Böhm C.,Schäfer U., „Analog Speicher," URL: https://www.analogspeicher.org,/

[7] Droste, M., Letellier, J., Böhm, C., Resch, T.: Combining High-Fidelity Visuals and Spatial Acoustics in Virtual Reality – Auralization of a Virtual String Quartet. Kultur und Informatik - Extended Reality, (2020), 179-192.

[8] Authenticity in Music Production Homepage, URL: https://www.fhnw.ch/de/forschung-und-dienstleistungen/musik/hochschule-fuer-musik/projekte/authenticity-in-music-production/

[9] Geier M., Ahrens J., Spors S.: The SoundScape Renderer, A unified spatial audio reproduction framework for arbitrary rendering methods, in 124th AES Convention, Amsterdam, (2008).

[10] Harker, A., Tremblay, P.A.: The HISS Tools Impulse Response Toolbox: Convolution for the Masses. ICMC 2012: Non-cochlear Sound. The International Computer Music Association, (2012), 148-155.

[11] Xiang P.: Git Repository earplug~, URL: https://github.com/pd-externals/earplug/

[12] Git Repository SOFALizer-for-pd, URL: https://github.com/sofacoustics/SOFAlizer-for-pd/

[13] Yadegari, S., Moore, F., Castle, H., Burr, A., Apel, T.: Real-Time Implementation of a General Model for Spatial Processing of Sounds, (2002).

[14] Gardner: W.: Efficient Convolution without Input/Output Delay. J. Audio Eng. Soc., vol. 43, no. 3, (1995), 127-136.

[15] Line6 Helix, URL: https://line6.com/helix/resources.html

[16] Line6 Resources, URL: https://l6c-acdn2.line6.net/data/6/0a020a3e357b58ff87bad59f0/application/pdf/helix-blog-what-is-an-ir.pdf

[17] NUX Optima Air, URL: https://www.nuxefx.com/optima-air.html

[18] Authenticity in Music Production, URL: https://www.fhnw.ch/en/about-fhnw/schools/school-of-engineering/institutes/research-projects/authenticity-in-music

[19] Princeton PT2399 Echo Processor, URL: http://www.princeton.com.tw/en-us/products/multimediaaudioic/echoprocessor.aspx

[20] Lindau, A., Brinkmann, F.: Perceptual evaluation of individual headphone compensation in binaural synthesis based on non-individual recordings. Journal of the Audio Engineering Society 60, (2012), 54-62.

[21] Immersive Audio Guiding System Homepage, URL: https://www.fhnw.ch/de/forschung-und-dienstleistungen/musik/hochschule-fuer-musik/projekte/immersive-audio-guiding-system-iags

[22] Resonance Audio: Unity, URL: https://resonance-audio.github.io/resonance-audio/develop/unity/getting-started.html

[23] Resonance Audio, URL: https://github.com/resonance-audio/resonance-audio

[24] Farina, A., Bellini, A., Armelloni, E.: Non-Linear Convolution: A New Approach for the Auralization of Distorting Systems. 110th AES Convention, Amsterdam, (2001).

[25] Reed, M., Hawksford, M.J.: Practical modelling of nonlinear audio systems using the Volterra series. 100th AES Convention, Copenhagen, (1996).

[26] Hélie, T.: On the Use of Volterra Series for Real-time Simulations of Weakly Nonlinear Analog Audio Devices: Application to the Moog Ladder Filter. 9th Conference on Digital Audio Effects, Montral, (2006).