DOI: 10.61782/fa.2023.1042

forum acusticum 2023

THE BINAURAL RENDERING TOOLBOX. A VIRTUAL LABORATORY FOR REPRODUCIBLE RESEARCH IN PSYCHOACOUSTICS

Daniel González-Toledo[1*] **Luis Molina-Tanco**[1] **María Cuevas-Rodríguez**[1]
Piotr Majdak[2] **Arcadio Reyes-Lecuona**[1]
[1]Telecommunication Research Institute (TELMA), Universidad de Málaga, Málaga, Spain
[2]Acoustics Research Institute, Austrian Academy of Sciences, Austria

ABSTRACT

The Binaural Rendering Toolbox (BRT) is a set of software libraries, applications, and definitions aimed as a virtual laboratory for psychoacoustic experimentation. The BRT is developed in the framework of the SONICOM project [1] and will include the algorithms developed in the 3D Tune-In Toolkit [2] in a new open, extensible architecture. At the core of the BRT Toolbox, a library provides C++ implementations of listener models, source models, and environment models, including a growing collection of portings to different audio frameworks such as PureData, MaxMSP and VST plugins, by means of the Avendish library. In addition, the BRT also includes an application controlled via the Open Sound Control (OSC) protocol.

This paper describes the architecture of the BRT, its main features, and its application to reproducible psychoacoustics experiments. The toolbox provides a complete trace of the experiment, including the delivered binaural audio, annotated with the listener and source movements. For this purpose, a new SOFA convention is proposed to store dynamic measurements, facilitating their use in the Auditory Model Toolbox (AMT).

Keywords: *binaural, audio rendering, psychoacoustics, auditory virtual reality.*

**Corresponding author*: *dgonzalezt@uma.es.*

[1] https://www.sonicom.eu/
[2] https://github.com/3DTune-In/3dti_AudioToolkit

1. INTRODUCTION

Immersive audio researchers have demanding needs for the open software development community, including individualised binaural simulation in dynamic settings, where multiple sound sources and multiple listeners move in virtual reverberating spaces, in real time. Individualisation is an important area of research where the maximum flexibility and support of updated standard formats for Head-Related Transfer Function (HRTF) files is also expected from software.

To the challenging demands on software performance we must add the variety of audio frameworks employed by researchers, which surely include standalone C/C++ developments, but also Pure Data, Max/MSP patches, Unity 3D, Unreal, VST3 or Web (HTML5) audio deployments. Of special interest is the ability to interface with the Auditory Modeling Toolbox, an Octave/Matlab library advancing the use of software models of many stages in the human auditory system [1]. Of course, different researchers work with different operative systems, which demands also for multi-platform approaches that allow deployment of software to the Windows, MacOS and Linux operating systems. The Open Sound Control (OSC) protocol is often used to connect different components and allow interoperability in this heterogeneous software ecosystem, which adds another interface requirement for new modules to be integrated with existing software.

The availability of open, version controlled software, and the reduction of the human factor (when possible) by the use of auditory models, contribute towards a higher degree of reproducibility in immersive audio research. However, research with human participants in dynamic, real-time virtual auditory spaces (VAS) is a challenge to replicability. Not only must the code to reproduce the stim-

uli during a experiment be made available to other researchers, but also all the variables involved in the interaction of the participants with the system, which must be captured and stored, ideally using standard open formats available to the research community.

The 3D Tune-In Audio Toolkit [2] (3DTI-Toolkit) is a set of software libraries which address some of these needs. It is composed of an open-source C++ library with examples, and partial ports to Unity, VST and Javascript/Webaudio. The libraries are version controlled and available on Github [3], together with the Binaural Test Application (BiTA) [4], which allows to interactively test all the capabilities of the 3DTI-Toolkit, including full real-time 3D binaural audio rendering, hearing loss and hearing aid simulations. BiTA is not open source, but is freely available for Windows, Mac and Linux. BiTA is mainly a graphical user interface to the 3DTI-Toolkit, but some of its functionality can be controlled via OSC commands, which allows for integration with other software, and for (partial) batch-mode execution of the application. BiTA allows to save the binaural audio to WAV format, and the static configuration of the parameters of the simulation at any moment. However, audio and parameters are saved separately. For the parameters, only snapshots of their values can be saved at the press of a GUI button. Their full trajectories in parameter space cannot be recovered after simulation.

The 3DTI-Toolkit implements the binaural rendering process chain in a specific manner. All stages in the chain can be switched on and off, and all the parameters of each stage can be fully controlled. Being open software, it can be modified to use different algorithms in any of the stages, but it was not designed with that feature in mind. Extending the Toolkit is not trivial and probably requires deep understanding of the source code.

This paper introduces the Binaural Rendering Toolbox (BRT). The BRT is a set of libraries, examples, interfaces and file formats, developed with all these needs in mind. Taking the well tested, efficient algorithms of the 3DTI-Toolkit, it goes beyond open software to *Open Architecture*: the internal interfaces between the different stages of the rendering chain are designed so as to allow easy development of new algorithms that extend the toolbox. The different stages are separable, which allows to port only specific stages to multiple platforms and audio hosts. Open Architecture, interoperability and reproducibility of research have guided the design of its architecture. In the reminder of the paper we present the architecture and the different components of the toolbox.

[3] https://github.com/3DTune-In

[4] https://github.com/3DTuneIn/3dti_AudioToolkit/releases/

2. ARCHITECTURE

The BRT is a set of tools for real-time binaural audio rendering, designed with an open architecture for researchers to build highly customised virtual acoustic scenarios. There are four main components of the BRT toolbox (see Figure 1):

1. At the core of the BRT is the **BRT library**, a header-only C++ library organised into modules that can be interconnected in different configurations, so as to allow maximum customisation. More detail is given in Section 2.1.

2. The **BRT Applications** are executables for different platforms, including wearable devices. The applications are interoperable audio renderers and auxiliary tools, such as motion trackers. The BRT allows to build renderers for specific scenarios, or generic renderers, configurable for different virtual scenarios. The BRT application (BeRTA), which is already under development, is an example of the latter.

3. The **BRT Portings** are deployments of the library modules to different rendering engines/hosts, such as PureData, Max/MSP or VST3.

4. Finally, the **BRT Definitions** are a set of common standards, such as an OSC syntax, a new SOFA convention for the capturing of data from psychoacoustic experiments [5], documentation, and open-source examples.

The components that form the BRT will allow us to achieve modularity at two levels: at compile-time and at run-time. At-compile time, C++ developers will have at their disposal different choices as to how to interconnect the library modules according to a desired functionality.

Run-time modularity will be achieved both by integrating the BRT Portings into different rendering frameworks (PureData, etc.) and by using the interoperable BRT applications. This will allow researchers to include the BRT algorithms in their existing experimental apparatus, while minimising the need for coding.

[5] Currently in the pre-proposal state, see section 3

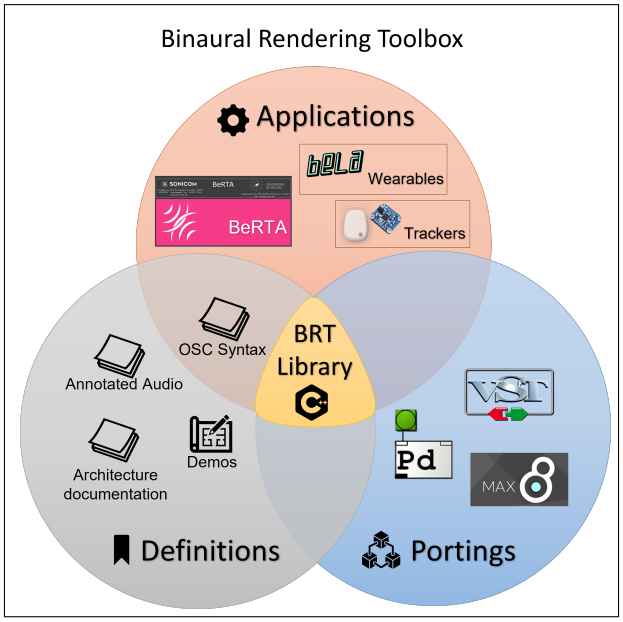

Figure 1. Binaural Rendering Toolbox conceptual diagram.

Finally, we are hoping that the combination of the BRT applications and the new SOFA convention for Annotated Audio (Section 3) will increase reproducibility of psychoacoustics experiments supported by the BRT and facilitate the connectivity with the Auditory Model Toolbox.

2.1 The BRT Library

The BRT library is a new open source C++ library [6] built on the experience gained with the 3DTI-Toolkit, a very efficient binaural rendering library, but difficult to customise and extend. Modularity and extensibility have been the main design principles for the BRT library. It is a header-only library, which simplifies integration in applications.

The library is organised into three layers:

- The bottom layer implements a template-based mechanism for the interconnection of modules, based on the observer design pattern [3].
- At the middle layer, the library is organised into low-level processing and service modules. Processing modules are in charge of signal processing: attenuators, convolvers, filters, etc. Service modules are auxiliary modules, such as HRTF managers, and SOFA file readers.
- At the top layer, high-level meta-modules interconnect the middle layer processing and service modules in specific ways that we call models. Some examples which are in our road-map include listener models, source models, environment models or hearing loss models.

This organisation allows different uses of the library for C++ developers. Models can be interconnected to create applications. For example, a practical anechoic binaural scenario could be formed by interconnecting a listener model and a source model. The middle layer provides building bricks to design new models, while the bottom layer allows to create new low-level processing or service modules.

The extensibility provided by this design will allow the BRT library to satisfy current needs in psychoacoustics by implementing new features not present in the 3DTI Toolkit, such as multiple simultaneous listeners, directionality of sound sources, or fast swapping of HRTFs at runtime.

3. ANNOTATED AUDIO

An important aspect in science is its reproducibility. The Auditory Modeling Toolbox (AMT) [4], an Octave/Matlab toolbox for the development of computational models of the human auditory system, has a strong focus on reproducible research. The AMT is a collective effort which involves validating and tuning the models' predictions against experiments performed with human participants.

One of the current lines of research in the AMT is the advancement of dynamic models for which the validation against experiments with humans can be challenging, as both the spatial configuration of the sound sources and participants can change within each experiment trial, dynamically affecting the binaural audio that the participants receive. When modelling such experiments, the rendered binaural audio resulting from the spatial configuration of the actual trajectories of both sound sources and listeners needs to be synchronised with the spatial configuration. The synchronous audio and spatial information, i.e., annotated audio, needs to be stored in a file, the format of which needs to be well-defined. Figure 2 shows the potential interaction between BRT and an AMT-based model

[6] https://github.com/GrupoDiana/BRTLibrary

using the annotated audio to model dynamic experiments. The annotated audio file aims at storing all the raw data during an experiment with participants to be reused for validation of new auditory models, or to reproduce the experiments.

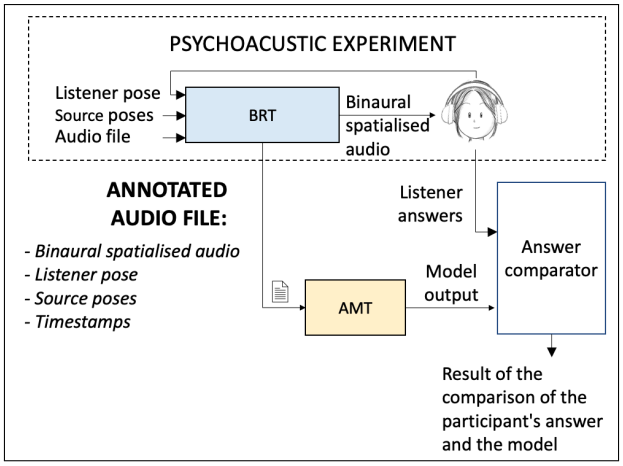

Figure 2. Potential interaction between BRT and an AMT-based model using the annotated audio file format.

We propose to incorporate the annotated audio as a new convention of SOFA, an established standard for sharing spatially oriented acoustic data. This helps ensuring its interoperability, ease of use and distribution. The SOFA standard, a file format for storing spatially oriented acoustic data, has been introduced in 2013 and is widely used by the community.

The working name for this new SOFA convention is ***SingleTrackedAudio***. While the convention is still in an early stage, relevant fields have been identified: position and orientation of listener head, position and orientation of sound sources, and rendered binaural audio samples. In detail:

- **Audio samples** are stored as a vector of SOFA dimensions *R x N* where *R* is the number of receivers (i.e., ears) and *N* is the number of audio samples per receiver.
- The SOFA dimension *M* represents the number of points in time with information about the spatial configuration of the listener and sound sources.
- The spatial information about the **listener** is stored in three matrices: *ListenerPosition*, *ListenerView*, and *ListenerUp*, with the first matrix describing the position and two latter matrices describing the orientation of the listener. Each matrix has the SOFA dimensions *M x C* where *C* is the number of spatial dimension and always three.
- Sound sources are represented by the SOFA emitters, i.e., the SOFA dimension E represents the number of sound sources.
- The spatial information about the **sound sources** is stored in three matrices: *EmitterPosition*, *EmitterView*, and *EmitterUp*, similar to the listener. Each matrix has the SOFA dimensions *E x C x M*.
- The link between the audio and the spatial configuration is provided in a vector called *M*. This vector contains the **timestamps of the measurements**: Each time the configuration is updated, a new entry is appended to the vector *M*. This allows the software to measure the listener tracking data and source positions at irregular time intervals, independently of the usually regular sampling of the audio.

Each SOFA file using the proposed convention is supposed to consider a single experiment trial, rather than long recordings of binaural audio. Additionally, the proposed convention considers simplifications, for example in cases where the listener or the sources are static. A proposal for the new convention describing the annotated audio will be submitted to the SOFA standardisation committee [5].

4. APPLICATIONS AND PORTINGS

In this section we describe BRT Applications and Portings being currently under development, the most important being BeRTA, the reference test application for the BRT Toolbox. Other applications are currently under development, such as a binaural renderer for the Bela Audio Board [7]. Initial portings of the first Listener Models to PureData, Max/MSP and the Steinberg VST3 Plugin SDK have also been developed to test the approach.

4.1 BRT Application: BeRTA

BeRTA is a real-time spatial audio rendering application. At the moment of writing, BeRTA is a single, standalone application with a high degree of configurability, that will

[7] https://bela.io

allow generation of different types of virtual sound scenarios. However, it might evolve into a series of different versions of the application, each one allowing for small configurations.

BeRTA integrates the BRT library for all audio signal processing and spatialisation, together with other open source third-party libraries to handle file formats (SOFA, WAV and others), to create the GUI, and to implement the OSC protocol. BeRTA is designed from the outset to be fully controlled via OSC. Thus, OSC is the primary interface to the application, whereas the GUI has almost exclusively informative tasks. Any other application or framework capable of producing OSC commands can control BeRTA. There is also an initial configuration file, which is loaded by BeRTA at startup to set default values for parameters, such as buffer size or sample rate.

The **GUI** in BerTA is a simple combination of terminal-like areas (see Figure 3) which provide information on the status of the application: the rendering parameters (what is enabled and what is not), or the parametres of audio sources and listener. Other terminals show the OSC commands received and sent, the commands executed by BeRTA, and the messages signaling the errors that BeRTA is programmed to detect.

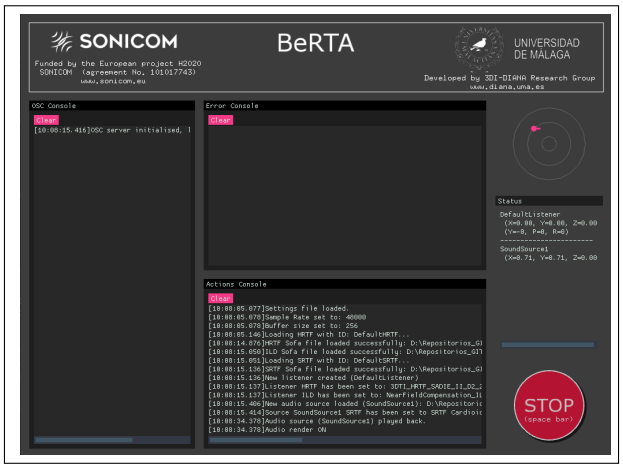

Figure 3. Screenshot of BeRTA v1.4.3

At the time of writing (v1.4.3), BeRTA has a modest list of OSC commands, shown in table 1 classified by groups. This OSC syntax is part of the BRT Definitions, and will be standard within the BRT Toolbox: other BRT applications will share the same syntax.

General messages refer to general commands to the system, such as making a general stop or pause, or deleting all existing virtual fonts in the simulation. These messages do not need a header *Control* messages are used to request information from the application, to initiate the connection or to close it. These messages should expect a response message back. *Source* messages correspond to actions on a specific sound source. Each source is created with a unique identifier, which must also be indicated by OSC at creation. Messages referring to *Listener* configuration are grouped in this type, in a similar way to those relating to the source.

Table 1. OSC messages by groups. The first column indicates the group and the header that must precede the command. The second column shows the different commands in the current version (1.3.0).

Type	OSC Messages (BeRTA v1.3.0)
General	/play, /pause, /stop, /loadHRTF /removeAllSources, /playAndRecord
Control (/control)	/connect, /disconnect, /ping, /version, /samplerate
Source (/source)	/loadSource, /removeSource, /play, /pause, /stop, /mute, /unmute /solo, /unsolo, /loop, /location /gain, /playAndRecord
Listener (/listener)	/location, /orientation, /setHRTF, /enableSpatialization, /enableInterpolation, /enableNearFieldEffect

4.2 Wearables

In the area of wearable devices and applications, work is being done in two areas:

- **BeRTA-mini**, a fully portable binaural renderer: A reduced version of BeRTA, BeRTA-mini is being developed for a wereable platform that combines a BELA, a specialised audio board capable of low latency audio and sensor data processing, a BN055 IMU sensor that integrates gyroscope, magnetometer and accelerometer, and a pair of headphones.
- **Interoperability for trackers**: Connectivity via OSC with MBIENTLAB's MetaMotionRL tracker is being developed as a first option. An application is being developed that will connect to the device

and translate the communication to OSC according to the syntax defined in BRT.

4.3 BRT Portings for multiplatform and multihost support

Mutliplatform and multihost [8] development can be seen as unrewarding tasks in Academia, but in the long term they support reproducibility of research and efficiency of the research community as a whole. We are currently evaluating two open source tools to support the double effort of multiplatform and multihost development of the BRT Toolbox : Avendish [6] and CMake [9] .

- **Avendish** [6] is an actively developed modern C++ library aimed precisely at helping audio software developers to reduce the development effort required to deploy M algorithms into N platforms, from $M \times N$ to $M + N$, with the effort quantified by N being provided for the most part by the developers of Avendish. At the moment of writing we have developed and tested proof-of-concept simple scenarios with some modules from the BRT library middle layer which have been deployed to PureData, Max/MSP and VST3.
- **CMake** is a well established build system to automatically generate build files for multiple platforms. Many modern, currently active C++ libraries use or have plans to use CMake as their only build system. Although still not the case for the BRT Library, for which we currently have manually written build files, there is already an experimental branch of the BRT library repository that uses CMake to integrate the BRT Library in Avendish.

5. DISCUSSION AND CONCLUSIONS

The BRT Toolbox is at early stages of development. Following the agile approach to engineering, the team developing BRT is looking for constant feedback from all the potential users and stakeholders involved. The first feedback was received internally within the consortium of the SONICOM European project, and greatly influenced the organisation of the toolbox.Our intention with this paper is to hopefully reach beyond SONICOM for testing and validation, so that we can better prioritise our development efforts.

User stories are another tool of agile engineering which is helping the BRT team to understand stakeholder needs, elicit requirements and prioritise development. For example, one of the active lines of research within the SONICOM consortium involves fast swapping of HRTFs, usage of auditory models, and preservation of the raw data in interactive experiments with subjects. These needs have driven the prioritisation of the listener models and the annotated audio format, which might become a new SOFA convention, and eventually, one of the results beyond the timespan of the SONICOM project.

At the time of writing this paper, the BRT Toolbox includes:

- The C++ code implementing a listener model with convolution based, anechoic binaural rendering, with multiple sound sources, near field simulation, and fast HRTF swapping.
- A Windows standalone version of BeRTA, a binaural audio rendering application controlled via OSC, which implements this listener model and saves all parameters of the interaction in annotated audio format.
- A wearable binaural renderer for the BELA.

The following components are currently in development, or planned for future iterations of development:

- Source models that support directivity of sound.
- Multi-listener rendering.
- Multiple reverberation models, including convolution-based, image source-based and hybrid models.
- Portings of the source, listener and reverberation models to PureData, Max/MSP and the VST3 plugin API.
- Hearing loss models.
- Hearing aid models.

[8] By *multiplatform* we mean developing the same application or library for different operative systems. For example, providing compiled or compilable versions of BeRTA (See section 4.1) for the Windows and the MacOS operative systems. By *multihost* we mean porting an algorithm to different audio host environments, such as PureData, Max/MSP or DAW plugin APIs such as Steingberg VST3.

[9] https://cmake.org

This list is given in no particular order related to priorities. A mixture of factors will continue to guide the BRT Toolbox roadmap. As the functionality of the BRT advances towards completion, a comparison in the context of similar tools should be made to assess the relative contributions of the BRT to the psychoacoustics research community.

6. ACKNOWLEDGMENTS

This study has been supported by SONICOM `https://www.sonicom.eu/`, a project funded by the European Union's Horizon 2020 research and innovation program under grant agreement No. 101017743, and the Spanish National Project SAVLab, under grant No. PID2019-107854GB-I00, funded by MCIN/AEI/10.13039/501100011033/FEDER, UE.

7. REFERENCES

[1] P. Majdak, C. Hollomey, and R. Baumgartner, "AMT 1.x: A toolbox for reproducible research in auditory modeling," *Acta Acustica*, vol. 6, p. 19, 2022.

[2] M. Cuevas-Rodríguez, L. Picinali, D. González-Toledo, C. Garre, E. de la Rubia-Cuestas, L. Molina-Tanco, and A. Reyes-Lecuona, "3D Tune-In Toolkit: An open-source library for real-time binaural spatialisation," *PLOS ONE*, vol. 14, p. e0211899, 3 2019.

[3] E. Gamma, R. Helm, R. Johnson, and J. M. Vlissides, "Design Patterns: Elements of Reusable Object-Oriented Software," p. 416, 11 1994.

[4] P. Majdak, C. Hollomey, and R. Baumgartner, "AMT 1.x: A toolbox for reproducible research in auditory modeling," *Acta Acustica*, vol. 6, 5 2022.

[5] P. Majdak, F. Zotter, F. Brinkmann, J. De Muynke, M. Mihocic, M. Noisternig, P. Majdak, A. Member, F. Zotter, F. Brinkmann, A. A. Member, J. De Muynke, and M. Mihocic, "Spatially Oriented Format for Acoustics 2.1: Introduction and Recent Advances," *Journal of the Audio Engineering Society*, vol. 70, pp. 565–584, 7 2022.

[6] J.-M. Celerier, "Rage Against The Glue: Beyond Run-Time Media Frameworks with Modern C++," in *Proceedings of the International Computer Music Conference (ICMC)*, (Limerick, Ireland), 2022.