1. SonData

Aiming to contextualize this document, I will firstly give a quick definition on data sonification and its interactive component.

Data Sonification

Kramer et al. (1999) defined data sonification as the "*the use of non-speech audio to convey information. More specifically, sonification is the transformation of data relations into perceived relations in an acoustic signal for the purposes of facilitating communication or interpretation*."

Interactive Data Sonification

Herman and Hunt (2004) defined Interactive Sonification as "*the discipline of data exploration by interactively manipulating the data's transformation into sound.*"

SonData is an Interactive Data Sonification toolkit, targeted at all practitioners interested in sonifying data. However, it also provides a set of tools that are useful to the academic and scientific data sonification communities. SonData provides tools for the interactive sonification of multidimensional datasets. In order to do so, it consists of a set of modules for both Parameter Based Sonification (PMSon) and Model-Based Sonification (MBS), offering interfaces like sonic scatter plots, data-driven 3D models, and a wide variety of sound synthesis modules.

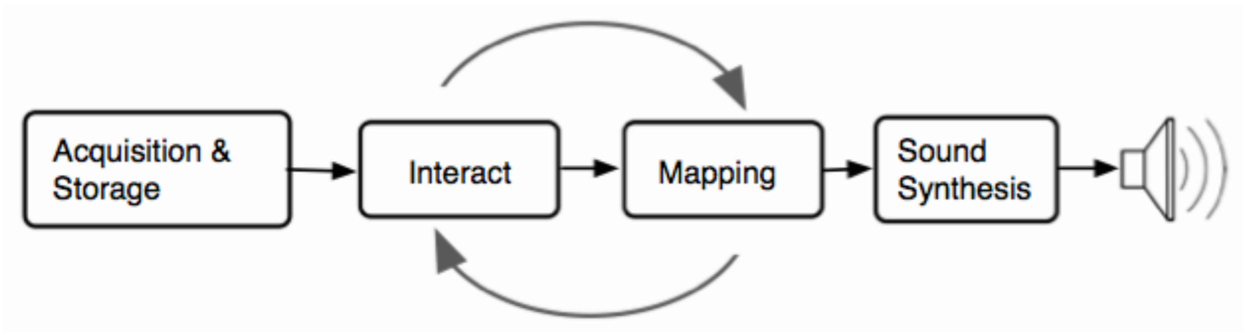
In addition to its academic properties, SonData can easily suit any project that deals with extensive data matrices, mapping processes or data-driven sound synthesis. It was developed in MaxMSP integrating the Jamoma framework, providing modularity, connectivity robustness and rapid prototyping in its application context.

2. How it works

As shown in table 2.1, sonData's modules are organized according to their function. Although SonData is a modular toolkit, the modules were designed to work according to a pre-defined hierarchy that provides the methodology for the development of a sonification system. Figure 2.2 aims to illustrate this methodology, where stored data is entered into the interaction modules. Most of the interaction with a sonification system is provided by the modules contained in the "interaction" and "mapping" categories. These categories would allow the user to apply the interaction component present in the PMSon and MBS. The outcome of this interaction is then mapped to sound synthesis parameters, thus providing the system's acoustic response and therefore closing the interaction loop.

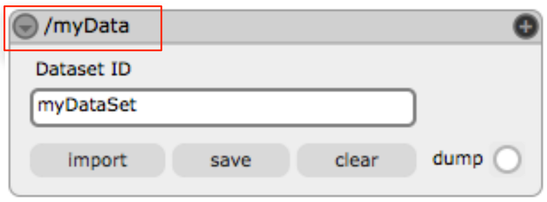
Acquisition and Storage	Interaction	Mapping	Sound Synthesis
• Jmod.son.table • Jmod.son.url	• Jmod.son.model • Jmod.son.brush • Jmod.son.Knearest • Jmod.son.mgraph • Jmod.son.graph	• Jmod.son.mapper • Jmod.son.mapperSingle • Jmod.son.linkModel	• Jmod.son.oscillator~ • Jmod.son.impulse~ • Jmod.son.subtractive~ • Jmod.son.waveshaper~ • Jmod.son.pulsar~ • Jmod.son.granular~ • Jmod.son.fof~ • Jmod.son.karplus~ • Jmod.son.resonator~

2.1 - Modules organized by category.



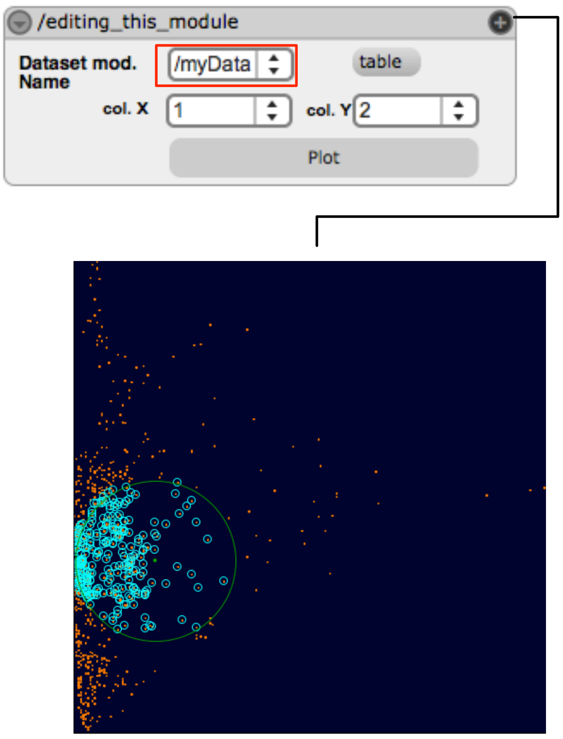
2.2 - Methodology proposed by SonData.

2.1 Interacting with data



2.4 - Dataset module and its ID (jmod.son.table)

The communication between the dataset and interaction modules is done remotely. The user must set a name for the module as well as an ID for the dataset (jmod.son.table; Fig. 2.4). The module's name will be shared with all the interaction modules. For example, [jmod.son.brush] (Fig. 2.5) is associated with the "myData" module. This means that, the plotted data belongs to dimensions 1 and 2 from the dataset contained in the [jmod.son.table] with the name **/myData** (Fig. 2.4). In the background, the interactions modules will get the **ID** of the selected [jmod.son.table]. This is a particularly useful function when there are more than one dataset and interaction modules on a given project, allowing the user to interact and easily switch between the available data.



2.5 - Module for sonic brushing (jmod.son.brush).

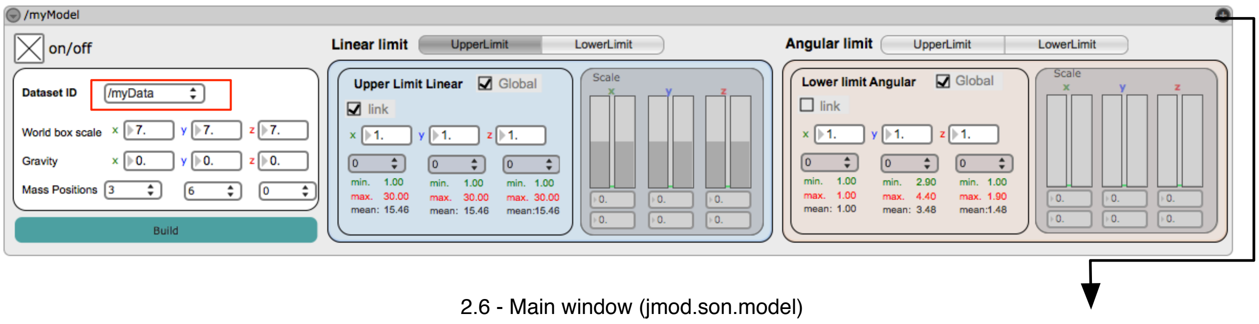
2.2 Building a sonification model

SonData provides modules to build a sonification model using the MBS paradigm. Hermann (2011) defined Model-Based Sonification (MBS) as "*the general term for all concrete sonification techniques that make use of dynamic models witch mathematically describe the evolution of a system in time, parameterize and configure them during initialization with the available data and offer interaction/excitation modes to the user as the interface to actively query sonic responses which depend systematically upon the temporal evolution model.*"

In sonData, the model can be built through the [jmod.son.model] (Fig. 2.6). This model provides a spring damping system driven by the dataset contained in [jmod.son.table] (Fig. 2.4). The module is divided into three sections:

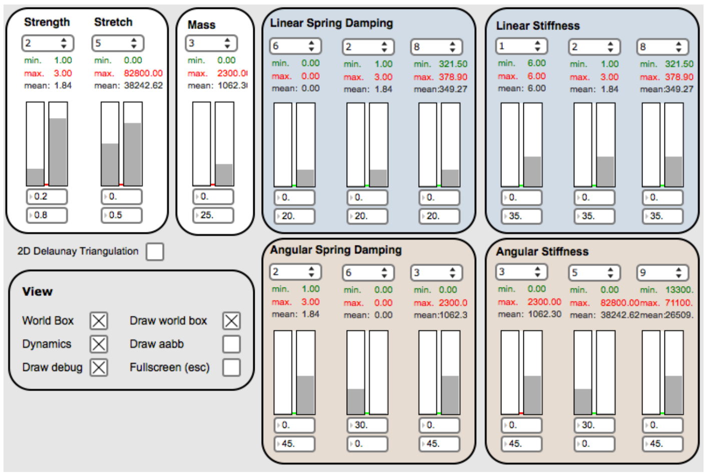
(a) - The main window (Fig.2.6), where the user can associate the model with the dataset ID, and set the model's setup. More precisely, the masse's position in 3D space, gravity, and the linear and angular spring boundaries allowed for each mass.
(b) - The Dynamics window (Fig. 2.7), allows the user to define the model's dynamics. This section provides access to parameters that define the model's behavior when excited. Parameters like masse's mass, spring stretch and strength, as well as linear and angular damping and stiffness for each spring.

(c) - The Model window (Fig. 2.8), where the user interacts with the model and sees how it behaves.



2.6 - Main window (jmod.son.model)

2.2.1 Dynamics window

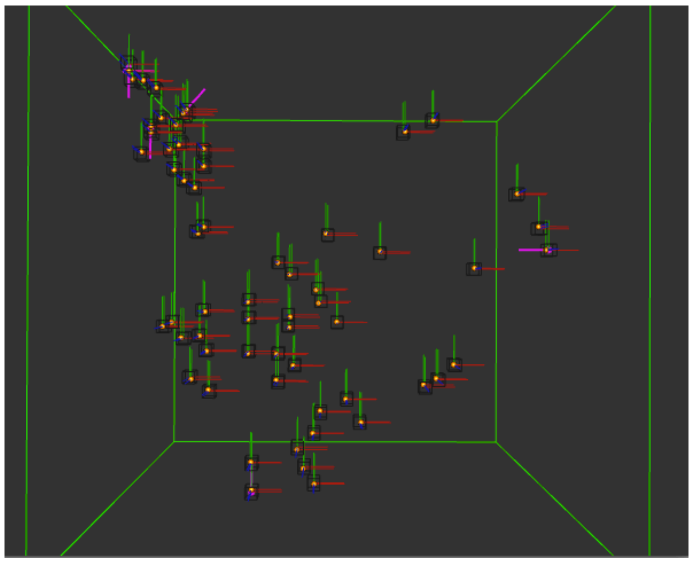


2.7 - Dynamics window (jmod.son.model)

As aforementioned, in the dynamics window the user can relate models dynamics with the underlying dataset. This is done by selecting one dimension (column) of the associated dataset, to the properties of each spring. Additionally, the user can adjust the scale factor for each property. This can be done in real time, thus allowing the user to obtain better performances in a short amount of time.

In sum, this window describes Hermann's aforementioned definition on MBS, more precisely: "*(...)dynamic models which mathematically describe the evolution of a system in time, parameterize and configure them during initialization with the available data(...)*".

2.2.2 Model window



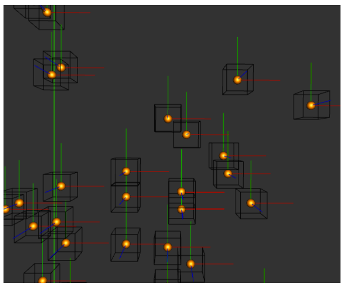
2.8 - Model window.

Here (Fig. 2.8), the user can interact with the model and see how it evolves in time. While in debug mode, the user can see world dimensions and the boundaries for each spring. Debug mode can be toggled in the dynamics window (Fig. 2.7). Note that the model is based on [jit.phys] objects.

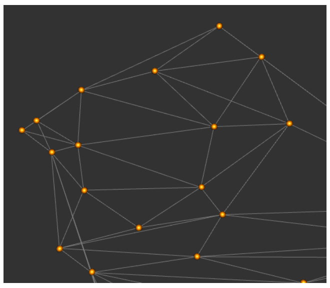
Another feature, is the option to perform a Delaunay triangulation within the masses.
As shown in Figure 2.10, the masses are attached to each other by links with their own stretch and strength properties. Thus, the energy introduced by the user's interaction, is transmitted to the rest of the system through the "Delaunay links", causing the model's structure to behave as whole.

Delaunay triangulation can be toggled in the dynamics window (Fig.2.7).

Note: At the moment, the Delaunay triangulation only performs in 2 dimensions, however, I'll hopefully be able to implement a 3D Delaunay in the near future.



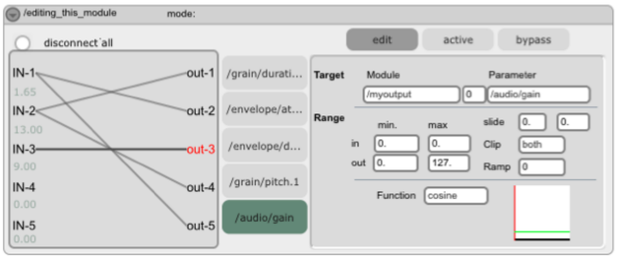
2.9 - Model window, closer view.



2.10 - Delaunay Triangulation.

3. Mapping

[Jmod.son.mapper] (Fig. 3.) provides the mapping between data output (as a result of interaction) and the sound synthesis modules. It can receive up to five values in a row and map them to sound synthesis parameters. Through a "connector" GUI, the user can easily switch between mappings, making it easier to experiment and hear different mapping strategies in a shorter amount of time. Additionally, the user can also set the polarity, ramp function and scale of the mapping.



Target Module /myoutput 0 Parameter /audio/gain
Range in min. 0. max 0.; out min. 0. max 127.
slide 0. 0.
Clip both
Ramp 0
Function cosine

3 - continuous mapper (jmod.son.mapper).

As in the previous modules, the mapping between incoming data and sound synthesis parameters is done remotely, in this case, through Open Sound Control (OSC). The user must choose the desired module and its parameter from the available menu, in this way mapped values are sent as an OSC message to the target module and respective parameter.

Note: this module is a continuous mapper and is suitable for use with PMSon. In the case of MBS, one would probably want to use the [jmod.son.linkModel].

4. Final notes

The current state of SonData provides a flexible basis to develop systems within the sonification domain. However, many of SonData modules are suitable for a wide variety of contexts that involve mapping processes and sound synthesis generation. Additionally, providing OSC communication with and within all SonData modules, is an important factor in both interactive sonification and artistic domains. For example, in the interactive sonification domain, users can easily connect any external controller (Wiimote, for instance) to interact with a sonification system, or map data to sound spatialization systems. On the artistic domain, one could like to receive data from a web feed and map it to sound synthesis parameters, as well as map incoming sensor data to a specific output with any software that provides OSC communication.

4.1 work in progress

SonData is a very recent project, thus there are some bugs and a lack of documentation. Beyond this, [jmod.son.modelLink] is at the top of to do's list. This module is responsible for the linkage between the sonification model and the sound synthesis parameters. While it's still able to play, there are some issue regarding voice addressing. It's a constant fight since all the synthesis modules should be changed when a major change is done. So, I think we can say that the sonification model [jmod.son.model] is pretty robust, but the linkage between its variables and sound generation is not.

Also, I'm currently working on a new sonification system that involves a virtual membrane whose dynamics are driven according to the underlying data.

4.2 Download and get involved - *(If you light a lamp for someone else it will also brighten you pat*(c)*h)*

In order to use SonData you should download the Jamoma framework. Note that you should use it with version 0.5.7 or earlier. SonData is still not fully compatible with the 0.6 alpha, which is a recent major change made in Jamoma foundations.

It would be really nice to have some feedback on SonData. There are lots of possibilities and sometimes it's not that easy to address them by myself. It would be really helpful to hear from you guys.

- www.github.com/joaomenezes/sonData
- www.cargocollective.com/joaomenezes/sondata

I'm looking forward to any suggestions, doubts or feature requests. You can reach me at joaommenezes@gmail.com. Also, if you want to get involved you can address some issues on the above mentioned SonData git repository.

Looking to further knowledge on sonification?

Take a look at the sonification handbook, it's freely available online.

http://sonification.de/handbook/

"This book gives a solid introduction to the field of auditory display, the techniques for sonification, suitable technologies for developing sonification algorithms, and the most promising application areas."

Bibliography:

Hermann, T. & Hunt, A. (2004). The Discipline of Interactive Sonification. Proceedings of the Int. Workshop on Interactive Sonification, Bielefeld, Germany

Hermann, T. (2011). Model-based sonification. In Hermann, T., Hunt, A., Neuhoff, J. G., editors, The Sonification
Handbook, chapter 16, pages 399–427. Logos Publishing House, Berlin, Germany.

Kramer, G, B Walker, T Bonebright, P Cook, J Flowers, N Miner, and Neuhoff. 1999. Sonification report: Status of the field and research agenda. *Prepared for the National Science Foundation by members of the International Community for Auditory Display*.