Motion capture studies of action-sound couplings in sonic interaction

STSM COST Action SID report
Alexander Refsum Jensenius
Affiliation: fourMs, University of Oslo, Norway
Host: TMH, KTH, Stockholm, Sweden

22 June - 3 July 2009

1 Background

Combining terminology from (Schaeffer, 1966) and (Cadoz, 1988), we may differentiate three different *action-sound types*, as presented in (Godøy, 2006): *impulsive*, *sustained* and *iterative*. As shown in Figure 1, each of the action-sound types may be identified from the energy profiles of both the action and the sound.

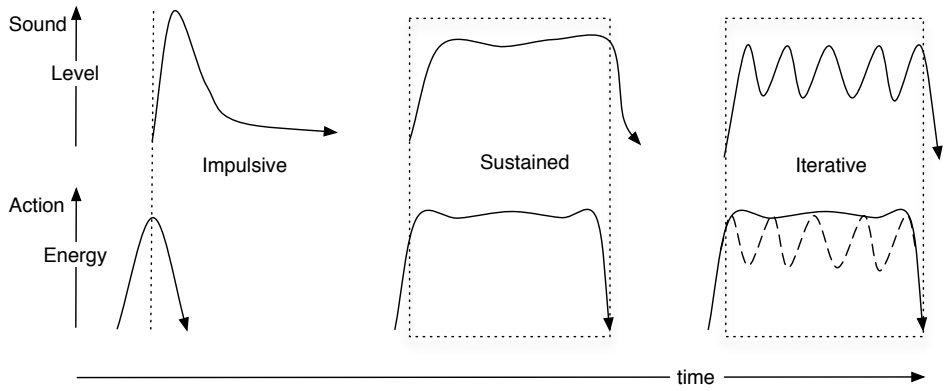

Figure 1: Sketch of action energy and sound levels for the three main types of sound-producing actions. The dotted lines/boxes suggest the duration of contact during excitation.

Getting more knowledge about the production and perception of such action-sound couplings is crucial for being able to create new artificial action-sound relationships in electronic devices. One method for obtaining such knowledge is to carry out studies where people are asked to move to sound.

In my previous research I have looked at air instrument performance (Godøy et al., 2006b), free dance to music (Casciato et al., 2005) and sound tracing (Godøy et al., 2006a). The latter is particularly interesting, since people are asked to move in a reduced movement space, i.e. typically with a digital pen, mouse or other human interface device. This allows for comparison of a few movement features to features in the musical sound, and may also open for reversing the process: to create sound through similar types of movement.

Throughout the years I have experienced a number of challenges when it comes to recording and storing data for such studies. Typically, we deal with data coming from different motion capture devices (Optitrack, Wii, Wacom, mouse, etc.), all which have different resolution and sampling rate. But we also want to store these data synchronised with related audio/video recordings and information about the setup, stimulus being played, etc. To handle synchronised streaming and storage of such data, we have suggested development of the *Gesture Description Interchange Format* (GDIF) (Jensenius et al., 2006). This is a structured approach to handling music-related movement data and media, and is building on other established standard (i.e. OSC, SDIF, XML). This STSM builds on previous work on recording similar types of data (Jensenius et al., 2008), and the work done by Kristian Nymoen during his SID STSM to McGill University in 2008 (Nymoen, 2008).

2 Objectives

The main objective of the STSM was to create a software solution for recording, playing back and analysing data/media for observation studies of action-sound couplings, and to carry out a small pilot study. This can be broken down into the following subgoals:

- Compare and exchange information about the software solutions developed at KTH and UiO for recording motion capture data synchronised to audio, MIDI and video data.
- Develop a software solution for streaming and storing multimodal data, based on OSC, XML and SDIF/GDIF.
- Test the above mentioned software in a pilot study where people would move spontaneously to a small set of action-sound couplings with different sonic qualities.

3 Results

Most of the STSM was spent on software development. A pilot study was conducted during the last day, but it was not sufficient time to carry out

any analysis on the material during the STSM. This section will therefore mainly focus on the software development.

3.1 Software platform

The graphical programming environment Max/MSP/Jitter was chosen for the software development. It could have been preferable with a free software solution, so that the software could easily be distributed to all SID partners. However, many of the partners already have access to Max, and Max also allows for creating standalone applications that can be shared and used without buying a software license.

The development further built on two frameworks for Max:

- FTM is a library developed at IRCAM, and allows for working with complex data structures in Max (Schnell et al., 2005). FTM also provides tools for writing and reading *Sound Description Interchange Format* (SDIF) files, a core technology used in the software solution being developed in the STSM.
- Jamoma is a modular framework for Max, consisting both of a structured approach to Max development, but also a large collection of modules and components (Place and Lossius, 2006). Most of the software development in the STSM has been made as Jamoma modules, and can therefore be easily reassembled to be used in various combinations.

Development was split into two parts:

- SID recorder
- SID player

These parts will be presented and discussed in the following sections. The patches are available in the UserLib of Jamoma, and can be checked out from Jamoma's repository (see www.jamoma.org for details).

3.2 SID Recorder

The first part of the setup is the *SID recorder*. This is a patch that allows for recording various types of data synchronised with audio and video. Figure 2 shows a screenshot from the patch, during test recordings with a Wacom tablet. In this section we will go through the functionality of the patch.

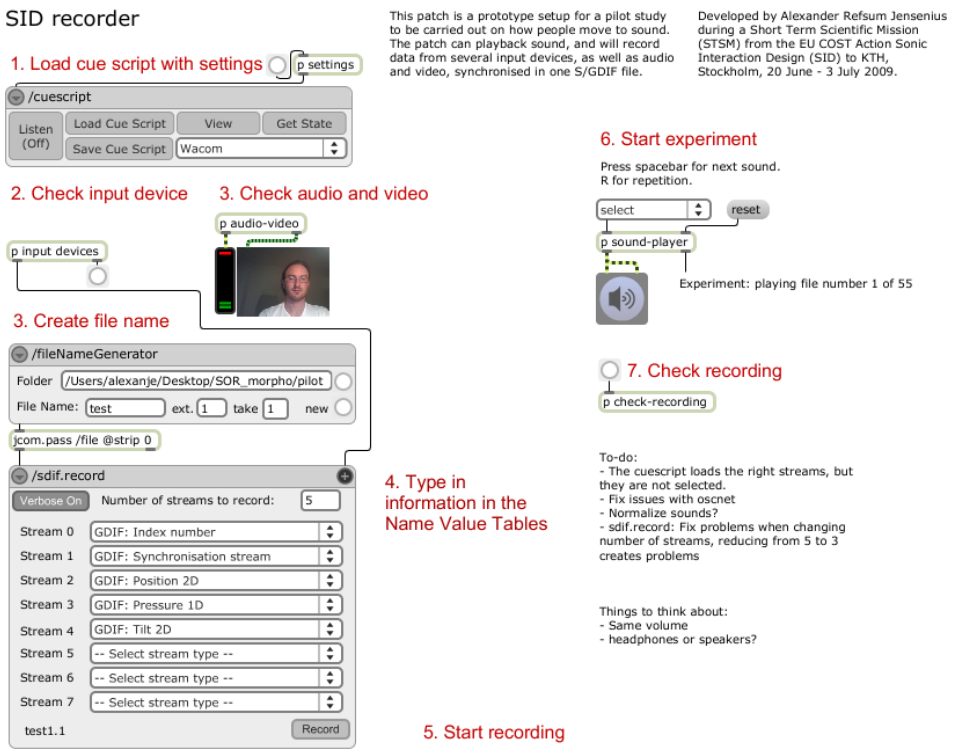

Figure 2: Screenshot from the patch *SID recorder*.

3.2.1 Cuescripts

One of the uses of Jamoma is the handling of music and video in various types of installations and performances. Here quick loading of different *cues*, i.e. named presets, is essential. For this reason there are several mechanisms built into Jamoma for handling cues, one being the **jmod.cuescript** module. This module allows for creating named presets that can easily be loaded and triggered as necessary, and it also provides a good mechanism for handling stages in an experimental setup.

In the current SID Recorder several cues have been implemented: *Initialization*, *Audio/video on*, *Wii*, *Wacom*, *Mouse*. The two first cues are used when opening the patch, so that the researcher can be sure that the patch is correctly initialized and that audio and video is turned on. The three last cues are used to select between the three different recording possibilities in the patch. More cues can easily be added if other input devices are going to be used. In our experience, using strictly defined cues saves a lot of time when carrying out experiments, and also ensures that everything is on and working.

3.2.2 Input devices

This subpatch contains modules for each of the input devices (**jmod.mouse**, **jmod.wacom**, **jmod.wii**), as can be seen in Figure 3. The modules are run into an abstraction called **sdif-pack** which formats the data so that they can recorded using the sdif recording objects in FTM.

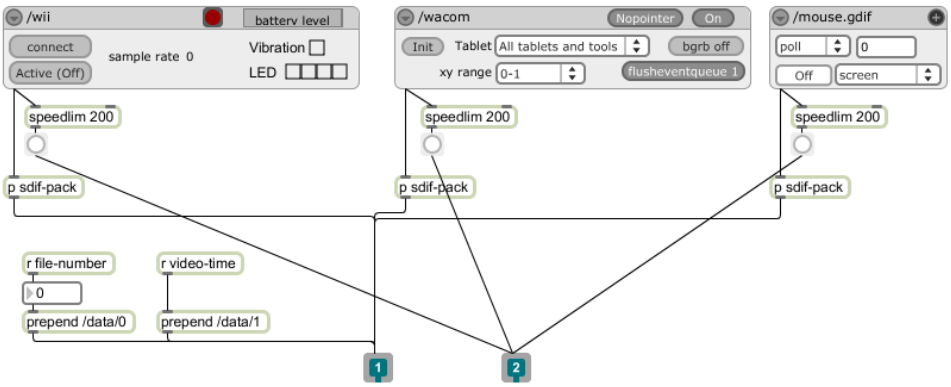

Figure 3: Subpatcher containing three input device modules.

3.2.3 File name generator

The **jmod.fileNameGenerator** module creates file names with incremental numbers (Figure 4). This is useful when carrying out experiments where

it is necessary to constantly save new files. It is possible to select a folder where files are stored and the root name and extension of the files. The module will then create files with incremental numbers.

Figure 4: The **jmod.fileNameGemerator** module generates incremental file names.

3.2.4 SDIF record

The **jmod.sdif.record** module encapsulates the **ftm.sdif.write** object, which is used for writing to an SDIF file. The module also handles setting up the number of streams to be recorded and the names and descriptions of the streams (Figure 5).

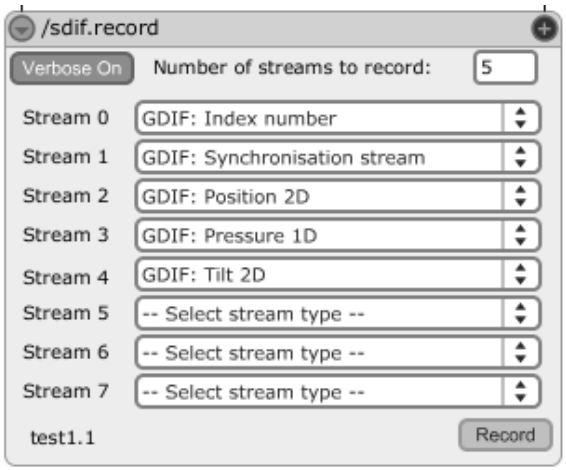

Figure 5: The **jmod.sdif.record** module.

By clicking on the icon in the upper right hand corner of the **jmod.sdif.record** module, a popup window allows for setting the *Name Value Tables* (NVTs) in the header of the SDIF file (Figure 6). This includes information about the date, location, experimental setup, author, etc. Keeping such metadata in the same file as the recorded data allows for easier sharing of data, since everything is included in one file.

Figure 6: Popup window where it is possible to type in information in the *Name Value Tables* (NVTs) of the recorded file.

In the jmod.sdif.record module it is also possible to select what types of streams are being recorded. We have previously defined specific GDIF types for each device (Jensenius et al., 2008), but have realized that it is more flexible if we come up with some generic descriptors. So for this setup I defined the following list of generic motion and position descriptors:

```
"-- Info --"
"GDIF: Index number" XIDX "index_number" "Index_number"
"GDIF: Synchronisation stream" XSNC "synchronisation_stream" "Sync"
"-- Sound descriptors --"
"SDIF: MIDI" 1MID "Status_byte\\\, First_data_byte\\\,
             Second_data_byte" "MIDI"
"-- Gesture descriptors --"
"GDIF: Position 1D" XPO1 "X" "1D_position"
"GDIF: Position 2D" XPO2 "X\\\, Y" "2D_position"
"GDIF: Position 3D" XPO3 "X\\\, Y\\\, Z" "3D_position"
"GDIF: Orientation 3D Euler" XEUL "yaw\\\, pitch\\\, roll" "3D_orientation"
"GDIF: Position & orientation 6D" XPOR "X\\\, Y\\\, Z\\\, azimuth\\\,
            elevation\\\, roll" "6D_Position-orientation"
"GDIF: Velocity 1D" XVE1 "X" "1D_velocity"
"GDIF: Velocity 2D" XVE2 "X\\\, Y" "2D_velocity"
"GDIF: Velocity 3D" XVE3 "X\\\, Y\\\, Z" "3D_velocity"
"GDIF: Acceleration 1D" XAC1 "X" "1D_acceleration"
"GDIF: Acceleration 2D" XAC2 "X\\\, Y" "2D_acceleration"
"GDIF: Acceleration 3D" XAC3 "X\\\, Y\\\, Z" "3D_acceleration"
"GDIF: Jerk 1D" XAC1 "X" "1D_jerk"
"GDIF: Jerk 2D" XAC2 "X\\\, Y" "2D_jerk"
"GDIF: Jerk 3D" XAC3 "X\\\, Y\\\, Z" "3D_jerk"
"GDIF: Pressure 1D" XPR1 "Pressure" "1D_pressure"
"GDIF: Tilt 2D" XTI2 "X\\\, Y" "2D_tilt"
"-- Biosignal descriptors --"
"GDIF: EMG" XEMG "emg_value" "EMG_data"
```

This list only contains some of the basic descriptors, most being in the

cartesian coordinate system, and should be expanded in the future.

3.2.5 Sound playback

The subpatch which handles playing back sounds during the experiment is built up of two copies of the **jmod.fileBrowser** module (Figure 7). Here it is possible to load a folder with files that will be played back randomly. Switching between the two modes can be done from the main patch.

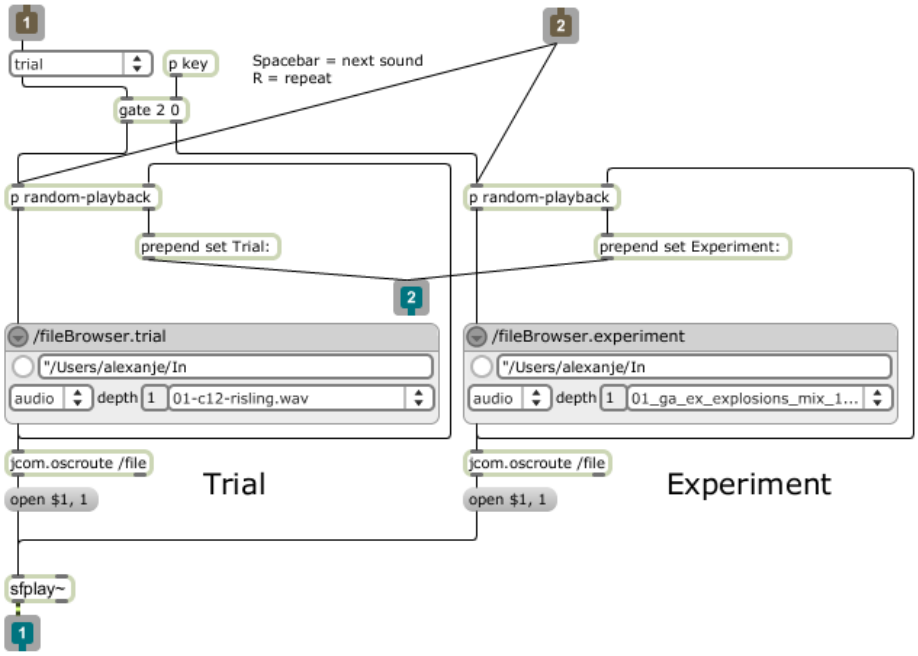

Figure 7: The subpatch containing the sound playback modules.

3.2.6 Check file

The last part of the SID Recorder patch is to check the recorded file after recording. This should be done to see that it has recorded properly, and to prevent data loss if the patch is reinitialized.

3.3 SID Player

Figure 8 shows a screenshot from the *SID player* patch. This patch allows for opening a recorded S/GDIF file with accompanying video file, and play back each stream independently. By connecting the index stream (IDX) to a sound playback engine it is possible to hear the original stimulus file being

played back. Similarly, the streams containing the position data can be used for drawing in the **lcd** object, as can be seen in the patch.

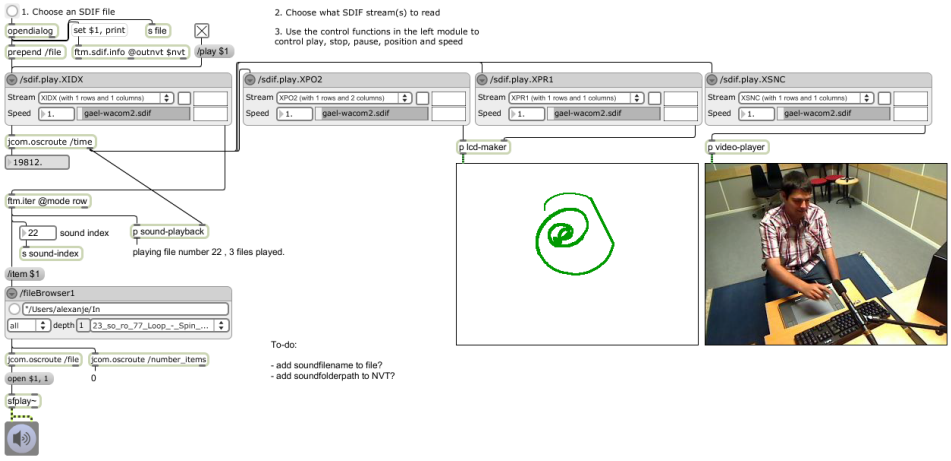

Figure 8: Screenshot from the patch *SID player*.

This player patch could also be extended with various types of analyses, graphing etc. However, due to time constraints this part was never finished.

4 Conclusions and Future work

This report has presented my work on development of a structured system for recording and playing back data and media from observation studies of people's movement to sound. The idea was to create a flexible system that can easily be expanded in the future. The system was tested extensively during my STSM, and I also carried out a small pilot study to see how the setup worked in a practical setting. Unfortunately, I did not have time to carry out an analysis on the recorded data.

Although the system seems to work well, there are many things that could be improved in the future:

- Develop more Jamoma modules for various input devices.
- Create a more coherent set of GDIF descriptors, both kinetic and kinematic.
- Develop modules for simple analysis of the data in Max, including saving the analysed data into separate streams in the data files.
- Develop analysis scripts in Matlab that can use the recorded data and media.
- Carry out more studies, using the above mentioned tools.

5 Acknowledgments

Thanks to Roberto Bresin for supervising my STSM. Also thanks to Gaël Dubus, Marco Fabiani, Anders Friberg, Kjetil Falkenberg Hansen, Kahl Hellmer and Giampiero Salvi for good lunches and interesting discussions, and Professor Sten Ternström for inviting me to the summer lunch in his house (including swimming in the pool!). Cudos to all the Jamoma developers for developing large parts of what the patches presented in this report have been built from, and similar credits to the FTM developers for their efforts.

References

Cadoz, C. (1988). Instrumental gesture and musical composition. In *Proceedings of the 1998 International Computer Music Conference*, Den Haag, Netherlands, pp. 60–73. 1

Casciato, C., A. R. Jensenius, and M. M. Wanderley (2005). Studying free dance movement to music. In *Proceedings of ESCOM 2005 Performance Matters! Conference*, Porto, Portugal. 2

Godøy, R. I. (2006). Gestural-sonorous objects: embodied extensions of Schaeffer's conceptual apparatus. *Organised Sound 11*(2), 149–157. 1

Godøy, R. I., E. Haga, and A. R. Jensenius (2006a). Exploring music-related gestures by sound-tracing - a preliminary study. In K. Ng (Ed.), *Proceedings of the COST287-ConGAS 2nd International Symposium on Gesture Interfaces for Multimedia Systems*, Leeds, pp. 27–33. 2

Godøy, R. I., E. Haga, and A. R. Jensenius (2006b). Playing 'air instruments': Mimicry of sound-producing gestures by novices and experts. In S. Gibet, N. Courty, and J.-F. Kamp (Eds.), *Gesture in Human-Computer Interaction and Simulation, GW 2005*, Volume LNAI 3881, pp. 256–267. Berlin: Springer-Verlag. 2

Jensenius, A. R., T. Kvifte, and R. I. Godøy (2006). Towards a gesture description interchange format. In N. Schnell, F. Bevilacqua, M. Lyons, and A. Tanaka (Eds.), *NIME '06: Proceedings of the 2006 International Conference on New Interfaces for Musical Expression*, Paris, pp. 176–179. Paris: IRCAM – Centre Pompidou. 2

Jensenius, A. R., K. Nymoen, and R. I. Godøy (2008). A multilayered GDIF-based setup for studying coarticulation in the movements of musicians. In *Proceedings of the 2008 International Computer Music Conference*, Belfast, pp. 743–746. 2, 7

Nymoen, K. (2008). A setup for synchronizing GDIF data using SDIF-files and FTM for Max. Cost action sonic interaction design (SID) short term scientific mission report (STSM), Input Devices and Music Interaction Laboratory (IDMIL), McGill University. 2

Place, T. and T. Lossius (2006). Jamoma: A modular standard for structuring patches in max. In *Proceedings of the 2006 International Computer Music Conference*, New Orleans, LA, pp. 143–146. San Francisco: ICMA. 3

Schaeffer, P. (1966). *Traité des objets musicaux*. Paris: Editions du Seuil. 1

Schnell, N., R. Borghesi, D. Schwarz, F. Bevilacqua, and R. Muller (2005). FTM — complex data structures for Max. In *Proceedings of the 2005 International Computer Music Conference*, Barcelona, Spain, pp. 9–12. 3