COST – Short-Term Scientific Mission, report
Action: E0601 – Sonic Interaction Design

Name: Kristian Nymoen
Institution: Musical Gestures Group, Department of Musicology, University of Oslo
E-mail: kristian.nymoen@imv.uio.no

STSM Title: A setup for synchronizing GDIF data using SDIF-files and FTM for Max.
STSM Host: Input Devices and Music Interaction Laboratory (IDMIL), McGill University

---

Preface

The purpose of this Short-Term Scientific Mission has been to investigate the use of the Sound Description Interchange Format (SDIF) as a container format for Gesture Description Interchange Format (GDIF) recordings.

Much of my work has been related to learning the equipment at the hosting laboratory, and to make the equipment capable of communicating with the software I have been using.

Background

In current music research, several types of devices for communicating data are widely used. These devices include motion capture systems, sensors for measuring pressure, acceleration, position, body-related data, etc., interfaces like keyboard, mouse and game controllers, digital musical instruments, video cameras and audio interfaces. Recording data from different sources presents a challenge in regard of synchronizing the data for post processing and analysis. This is mostly due to different standards for storing data and different sample rates in the various data streams [Jensenius et al. 2007]. Several standards present possible solutions to synchronisation issues, but they are often not meant for music research, and lack proper implementation of music-related data streams, such as audio, OSC, midi, etc. [ibid.].

As suggested in [Jensenius 2007: 213] the SDIF format can provide the necessary framework for GDIF recordings. A small pilot study for a project on co-articulation in piano performance was carried out at the Department of Musicology at the University of Oslo in December 2007 [Jensenius et al. 2008]. This study verified that an approach of recording movement-related data to SDIF files could be useful, especially so in regard of synchronizing data from multiple sources.

SDIF

The Sound Description Interchange Format has been developed at IRCAM[1], CNMAT[2], and IUA-UPF[3] as a standard for storing different sound representations [Wright et al. 1998]. SDIF contains different sound descriptions in separate streams along a common timeline. The streams consist of time-tagged frames where each frame contains one or more 2D matrices which contain the actual data. Since the frames are time-tagged, they do not rely on a specific sample rate,[4] and may exist anywhere along the timeline of the SDIF file, like illustrated in figure 1. The SDIF-specification includes a library of predefined stream (frame and matrix) types, like for instance the Picked Spectral Peak (1PIC) denoting peaks in the sound spectrum, where each 1PIC frame contains one 1PIC matrix consisting of 4 columns referring to amplitude, frequency, phase and confidence, and matrix rows corresponding to the number of peaks [Wright et al 1999]. The SDIF format is not constrained to the predefined stream types, but allows for other stream types to be defined in the header of each SDIF file. These stream types are defined with a four character frame ID and one or several four character matrix ID(s). The self-defined IDs starts with an X (e.g. **X**POS in my first example below). Frame types include the frame ID, frame description and matrix types to be included in the frame. The matrix type definition include the matrix ID and column descriptions. Matrix ID is often the same as Frame ID when there is only one matrix in each frame. The SDIF format also allows for writing meta information (e.g. date, name, descriptions of the file, etc.) in name-value tables (NVT) as part of the file header, this is especially useful for writing information on the recording setup in a GDIF recording.

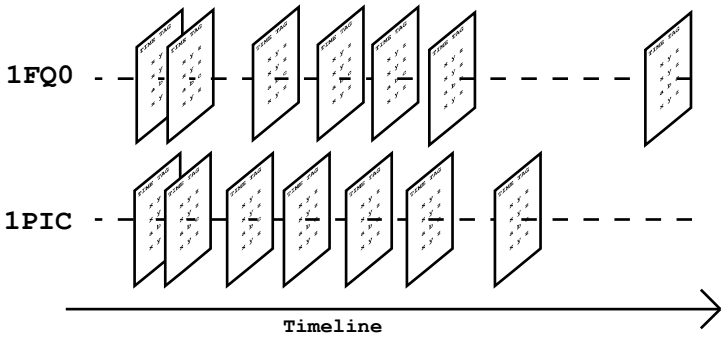

Figure 1: *Example of frames without a defined sample rate in two SDIF streams*

GDIF

The Gesture Description Interchange Format is currently under development as a standard for recording, streaming and describing musical movement. GDIF development is still mainly focusing on what aspects of musical movement to include in the format, and currently there is no defined format in which to store the files, but XML, SDIF and OSC has been suggested as protocols for

1 Institut de Recherche et Coordination Acoustique/Musique, Centre Pompidou, http://www.ircam.fr/
2 Center for Music & Audio Technologies, University of California at Berkeley, http://cnmat.berkeley.edu/
3 Institut Universitari de l'Audiovisual, Universitat Pompeu Fabra, http://www.iua.upf.es/
4 Other than the limitations in the double precision float denoting the time tag.
See http://recherche.ircam.fr/equipes/analyse-synthese/sdif/standard/sdif-standard.pdf for low-level structure details on SDIF.

storing and communicating GDIF data [Jensenius et al. 2007b]. One of the goals for GDIF development is to develop methods for structuring data from musical performances. These methods should be transparent of mediation-technology and independent of the interfaces being used for gathering data, i.e. capable of handling different/varying sample rates, bit resolutions, etc. simultaneously.

Software development

To provide a good user interface for recording and playing back SDIF files I have developed a set of Jamoma[5] modules. Jamoma is a standard for building high level modules for Max/MSP/Jitter[6], providing standards for user interface, communication and patcher structure [Place and Lossius 2006]. The modules developed for use with the SDIF format are to a large extent based on the SDIF objects in the FTM[7] library. The modules are:

- **jmod.sdif.play** – streaming data from SDIF files
- **jmod.sdif.record** – recording data to SDIF files
- **jmod.sdif.record.nvt** – module for writing name value tables to the SDIF file
- **jmod.sdif.record.control** – module for recording to predefined stream types
- **jmod.new_file_player** – module for generating indexed file names, useful for making several succeeding recordings within a single session.

In addition to these modules, I made the component jcom.sdif.record.data for converting lists of data to fmat[8] matrices, which is the format needed by jmod.sdif.record. The modules presented are still being improved, and because some functions are likely to change I do not include any instructions for module use in this paper. The help patches and html documentation files provided with Jamoma are continuously updated, and provide the necessary user instructions for the modules. The modules are available from the current active-branch of Jamoma.[9,10]

Setup in the IDMIL

The following equipment was used in my test recordings in the IDMIL.

5 http://www.jamoma.org
6 http://www.cycling74.com
7 http://ftm.ircam.fr
8 Float matrix FTM object.
9 https://jamoma.svn.sourceforge.net/svnroot/jamoma/branches/active
See http://www.jamoma.org/wiki for instructions on how to download.
10 Please note that the modules have only been tested on Windows. Testing and adjusting the modules for Mac use will be done in the near future.

Vicon V-460[11]

The Vicon-460 generates 3D models of objects based on position-tracking of reflective markers placed on the object and kinaesthetic models of the objects in question. 6 IR-cameras are recording the reflections of the markers, as a basis for calculating 3D positions for all the markers.

Tarsus is a Vicon application for for real time streaming of motion capture data. The data output from Tarsus is based on predefined models of the objects to be recorded. Based on the marker placement, the Vicon system recognizes a predefined object, e.g. a limb or a musical instrument, and outputs 3D position data and 3D orientation data for the objects. Tarsus works as a TCP server, communicating on port 800. Any TCP-client may connect to this server through the TCP/IP protocol and request for a single data packet or streaming of data packets.

Streaming data from Vicon to Max/MSP is not possible using the included objects for TCP/IP communication.[12] The mxj net.tcp.send object gives a message when the reply from the server is received, but they do not display the contents of the reply-package, which is where the Vicon data is located. To get the Vicon data to Max/MSP, it is therefore necessary to use a third party external or standalone software that is able to communicate with both Max/MSP and Tarsus. For this project I did try several 3rd party clients,[13] and found the Windows application QVicon-2OSC[14] by Christopher Frauenberger and Johannes Zmoelnig to be the most useful. This application provides a good user interface, and sends the parsed Vicon data as Open Sound Control messages through UDP. I have made a Jamoma module called **jmod.QVicon2OSC** with which one can remotely control the QVicon2OSC application, and receive data in Open Sound Control format. The use of this module makes it possible for a user to control all the data from within the Max/MSP environment without having to swap between different programs.

Polhemus Liberty

The Polhemus Liberty[15] uses an electromagnetic field (EMF) within which it can determine the absolute position and orientation of several sensors. The unit at the Input Devices and Music Interaction Laboratory has eight sensors for which the system outputs six degrees of freedom data:

3 DOF position data. *X, Y, Z*

3 DOF orientation data. *Azimuth, Elevation, Roll*

The Polhemus Liberty is capable of outputting data at a rate of 240 Hz. However, due to a previous

11 The information on the Vicon system presented here is based on information found in the Vicon manuals. These are part of the software package delivered with the Vicon system. In particular the documents I have been using are: ViconiQ_Option_RTE.pdf, Tarsus Communication.html, CommunicationsProtocol.rtf and ChannelSpecifications.rtf

12 Such as the mxj-objects net.tcp.send and net.tcp.recv.

13 Like tcpclient for PureData

14 http://sonenvir.at/downloads/qvicon2osc

15 www.polhemus.com

setup of the Polhemus system in the IDMIL, the data is output at a low, variable sample rate (30-35 Hz) in the recordings presented here.

Audio/Video

A Canon GL2 video camera was used for the video recordings, and all audio recordings was done in locally on the computer generating the audio using Max/MSP and PureData. In the setups presented here, synchronization issues related to the audio and video recordings is solved by recording 1 to a SDIF stream at the time the audio/video recording starts, and a 0 when it ends. This approach has worked in my setup, but is not the most reliable, and should be improved by e.g. recording several pulses or a ramp every *n*th sample.

T-Stick

The T-Stick is a family of musical instruments, described in [Malloch and Wanderley 2007]. It uses several capacitive sensors, accelerometers, pressure sensors and a piezoelectric crystal contact microphone as sensor inputs. The large amount of sensors give a complex structure for musical control of the sound module. The instrument is sensitive to tilt angles, shaking, bending, touch position, pressure and structural vibrations in the instrument.

Phantom Omni

The Phantom Omni[16] haptic device is a device for measuring position and orientation in six degrees of freedom (X, Y, Z, azimuth, elevation, roll). The device applies force feedback in three degrees of freedom (X, Y, Z). In the setup presented here, the device communicates through an experimental feature of DIMPLE [Sinclair and Wanderley 2007].

Recordings

To investigate the use of SDIF files as a container format for GDIF, I made recordings of improvised musical performances using the musical instruments, devices and motion tracking systems mentioned above. Two different recording setups were used. A local network was set up to allow data processing to be shared between several computers.

The recordings presented here are mostly for illustrative purposes to present the method in question, and thus I will not make any thorough analysis on the recorded data. I will however present visualisations of the recordings. Some of the patches, and files presented here are available from http://folk.uio.no/krisny/research/ .

16 http://www.sensable.com/haptic-phantom-omni.htm

Force-feedback algorithm with audio synthesis

In the first setup, a Phantom Omni haptics device was used along with a height map geometric texture algorithm made by Steven Sinclair. The force vectors to the feedback device was calculated based on the normal of the height map. The lateral surface forces that is felt by the user was applied to a synthesis model. We recorded the synthesized audio, the X, Y and Z position coordinates from the Phantom Omni as well as the X, Y and Z force coordinates. All processing directly related to the haptics software and hardware, as well as audio recording was done on a Windows PC. Video recording was done on a Mac Pro, and the recording of the GDIF data was done on another Windows PC. Data was sent over a local network as Open Sound Control messages. For efficiency reasons, all data, except for the sync streams, was recorded in 5-sample vectors, with a sample for each ms. Each sample in the vector was recorded to a separate SDIF-matrix row.

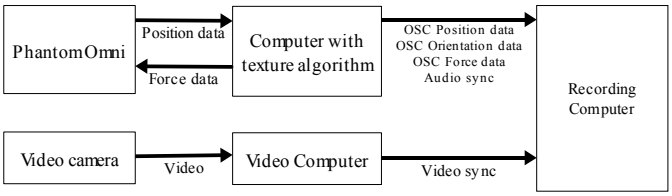

Figure 2: *Computer and device setup for the haptic device recording*

Five SDIF streams were defined for recording the data:[17]

Position stream: Data stored in XPOS frames, each including a single XPOS matrix. The matrix columns refer to the three cartesian input position coordinates from the Phantom omni.

Force stream: consisting of three-column, five-row XFOR matrices within XFOR frames. The X, Y and Z columns make up the force vector for each sample. The matrix rows refer to five succeeding force vector samples.

Orientation stream: consisting of XOR9 frames with XOR9 matrices. Each XOR9 matrix consists of 5 rows, where each row contains one rotation matrix (3×3 matrix represented as 1×9).

Sync streams: One audio synchronisation stream consisting of XASY frames with a single-number XASY matrix. And a similar stream for video with XVSY stream and matrix. A sync pulse was recorded to the respective sync stream at the moment the recording was initiated.

Using the jmod.sdif.play module, the force-feedback stream can be played back to the Phantom Omni along with the audio and video files. This way it could be possible to do empirical studies with the exact same data. For instance, several users may experience the exact same haptic feedback and evaluate the sound from different audio synthesis algorithms.

17 The stream types in these recordings are basically the same stream definitions as purposed in [Jensenius et al. 2008], with some modifications as outlined in the text.

T-Stick

The second setup was done in two equal sessions. These were both recordings of improvised T-Stick performances. The devices used in this setup was a tenor T-Stick, Vicon 460 motion capture system capturing five markers on the T-Stick and four markers on the performer's head, a Polhemus Liberty with two sensors placed on the left and the right wrists as well as audio and video recording. The devices and computers in use are shown in figure 3. Audio/Video sync, Polhemus, and T-Stick data was sent using Open Sound Control. The Vicon Workstation communicated over TCP/IP with the QVicon2OSC application on the recording laptop. For this setup, the following SDIF streams were defined:

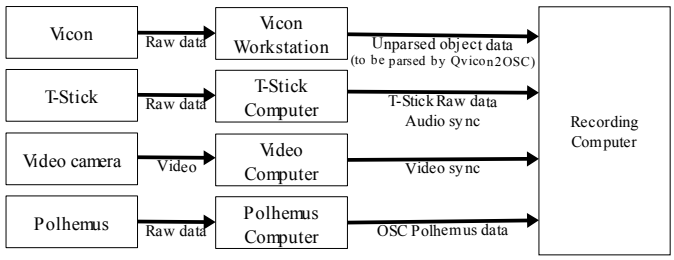

Figure 3: *T-Stick GDIF recording, computer setup*

Polhemus stream: Data stored in XPOR (position-orientation) frames, each including a single XPOR matrix. In each matrix there are six columns, referring to X, Y and Z position coordinates and azimuth, elevation and roll orientation coordinates. Each sensor is represented in a separate matrix row. The stream sample rate is variable: between 30 Hz and 35 Hz.

Vicon stream: For this setup, only the position and orientation of the two predefined objects (the head of the performer and the T-Stick) was recorded. These data may as well have been recorded to XPOR frames, but this would have caused conflicts with the XPOR Polhemus stream. Thus, the Vicon data was recorded to XVIC frames each containing a XVIC matrix. The columns in the XVIC matrix refer to angle-axis orientation coordinates, and X, Y and Z position coordinates. The two objects (T-Stick and head) are recorded to separate matrix rows. The Tarsus engine was set to operate at 250 Hz.

T-Stick stream: The T-Stick data was recorded as raw data to single-row fourteen-columns XTST matrices and XTST frames. The fourteen columns refer to different sensor data.

Sync stream: As in the first setup, a sync impulse was recorded when A/V recording started. This stream contains of XSYN frames and XSYN matrices. The audio and video files were recorded on one computer, and a synchronize signal was sent to the computer recording the SDIF file. I have chosen not to use the XAVS identificator as previously suggested [Jensenius et al. 2008], because it is likely that a synchronisation stream could refer to other things than only audio and video.

I set up a simple animation based on the data from the Polhemus and the Vicon data.[18] The axis from the Polhemus and the Vicon data was almost aligned, but not perfectly, making the animation a bit oblique. The position of the Polhemus EMF source is the origin (X=0, Y=0, Z=0) in the Polhemus data coordinate system, while the recorded Vicon data is offset compared to this. Thus I had to adjust the coordinates according to the origin position and the output data (Vicon outputs position data in millimetres, and the Polhemus in centimetres). I had some difficulties adjusting the orientation data,[19] so the orientation in the animation is incorrectly displayed. Orientation data is different in the two systems: Polhemus outputs azimuth, elevation and roll coordinates in degrees and the QVicon2OSC application outputs orientation data in radians in axis-angle format. Figure 4 shows the 3D animation placed in front of the video recording. This is played along with the audio recording. In the figure, the red ball (partially hidden) is the Polhemus marker on the left wrist, the green ball is the Polhemus marker on the right wrist, the blue cylinder is the T-Stick and the yellow egg-shape is the head, the two latter based on the Vicon position (and orientation) data. I have included the orientation, even though it is incorrectly displayed, because in spite of the error it provides some idea of the T-Stick behaviour. Along with this, one may also play back the T-Stick data, which in this animation could display the different control actions performed on the T-Stick (e.g. by illuminating the touch position).

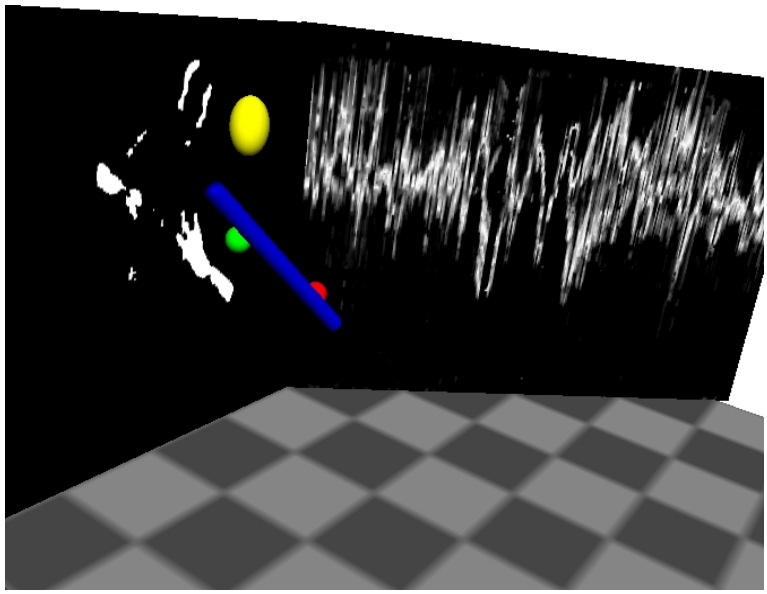

Figure 4: *Visualization of the recorded data along with video analysis. Motiongram and motion image technique from (Jensenius 2007).*

Evaluation of device synchronization using the Jamoma SDIF modules

To get an impression of the stability of the SDIF modules, I made a simple recording of two SDIF streams with 100 Hz sample rate and 2 ms offset between data in the two streams. Data was generated as random integers locally on the recording computer. This recording gave a perfect synchron-

18 A screenshot from this animation is shown in figure 4. The animation is done in Jitter using OpenGL, and is a modification of Polhemus Tools developed by Alexander Refsum Jensenius (Jensenius 2007). When playing back GDIF data along with the video, the jmod.sdif.play module is slowed down. Thus, to make the animations run smoothly, the video was played on one computer and GDIF data was sent from another computer through UDP. The animations show some delay between animation and video; the use of several computers for playback may be the cause of this. The sdif.play module will be improved to avoid the need for several computers when streaming video.

19 The difficulties regarding orientation data from the Vicon system is due to different variants of the axis-angle orientation coordinates between the Vicon system and the jit.gl.handle object that was used for the animation. The orientation axis in the two systems are not aligned.

isation, as displayed in figure 5 where all the samples are exactly 2 and 8 ms apart. Thus, the jmod.sdif.record module seems to be stable.

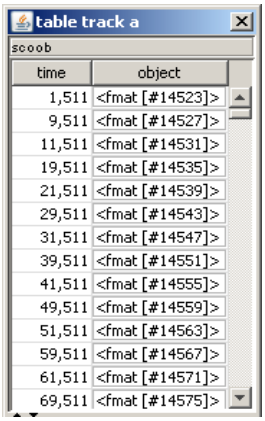

Figure 5: *First frames of recorded SDIF file, displayed in FTM track object*

The synchronisation issue is different when working on a network based setup, and a similar evaluation should be made for the setups presented above. In the first setup, data packets were sent at a constant rate of 200 Hz, and thus the time interval between packets of the same type should be 5 ms. However, the time intervals between the samples in the recorded file are not constant. The ping time between the machines was measured to 0 ms, but the large amount of GDIF data being sent did probably slow the network further down. The histogram in figure 6 shows this time flutter. This flutter is not necessarily critical when we are dealing with movement data due to a rather small sample rate, but would become a problem if we want to send audio data over the network.

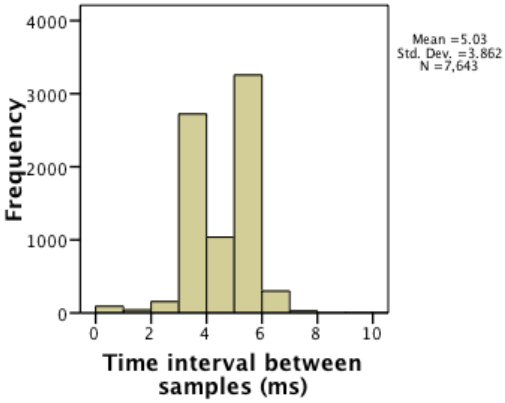

Figure 6: *Histogram of time interval between force-packets*

Network lag is also the most likely reason for the delay between video and animated data in the videos from these recordings presented on http://folk.uio.no/krisny.

In the second setup, the Polhemus (left and right wrist) and Vicon/accelerometer (T-Stick) data are data describing different objects which are independent of each other. Unfortunately, I did not have the time to make recordings where the Vicon system and Polhemus sensors both related to the same object, and thus I do not have the best data for evaluation of synchronisation between the different data sources. However, because the hands of the performer are on the T-Stick, the wrist data and the T-Stick data does in many cases refer to the same action trajectory. As an evaluation of the synchronisation of different data sources, I have chosen to look at these data. Figure 7 shows a Max/MSP multislider plot of the derivative of the Vicon Z-axis values

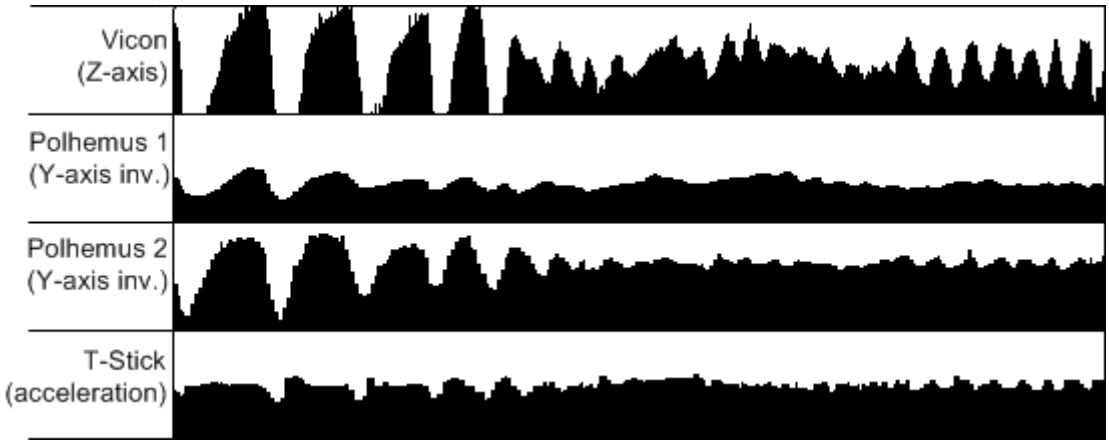

Figure 7: *Max/MSP multislider plot of a small segment (~7 seconds) of the T-Stick recordings. The devices were connected to three different computers, recording was made on a forth computer. The three top plots display up-down velocity, the bottom plot displays total acceleration (absolute value).*

for the T-Stick, the equivalent values for the Polhemus sensors and the magnitude of the acceleration vector from one of the T-Stick accelerometers. The values are scaled, and are not comparable in terms of amplitude. The Polhemus data and the T-Stick data are of lower sample rates than the Vicon data. These data are upsampled to the Vicon sample rate (with simple, constant interpolation) and so the time-axis should be comparable for all four data sources.

Measuring time between zero-crossings in the same direction of the Polhemus and Vicon data plotted above (before scaling) will give an indication of the level of synchronisation. Values greater than 0 is an upwards movement and less than 0 is a downwards movement. By looking only at the time differences between movements that obviously are the same movement (like the ones displayed in figure 7) I found time differences to be between approximately 5 and 176 ms. A histogram of the 79 measurements made is displayed in figure 8. Taking into consideration that the time interval between Polhemus samples is approximately 32 ms, and that the plots above refer to separate objects that do not necessarily move at the same time, the time differences are acceptable, but due to the insecure nature of these data, it is not possible to draw any hard conclusions on this experiment.

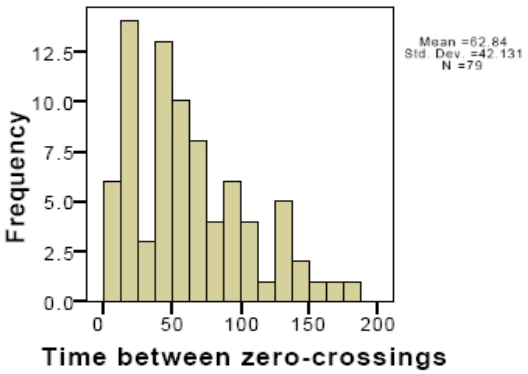

Figure 8: *Histogram displaying time intervals between Polhemus (right hand) and Vicon (T-Stick) data zero-crossings.*

The T-Stick recording was done in two sessions, where the Vicon system in both cases was set to operate at 250 Hz. For unknown reasons, the Vicon data in the first session was only recorded at a variable sample rate of approximately 120-130 Hz. In the second session the average sample rate for the recorded Vicon stream is 250 Hz, but there is time flutter similar to the time flutter in the force stream as presented above. Figure 9 shows a histogram of the time interval between Vicon samples in one of the recordings from the second T-Stick session. I assume this flutter is mostly due to the amount of data being sent over the network. A lower Vicon sample rate or a faster network connection may possibly have solved this problem.

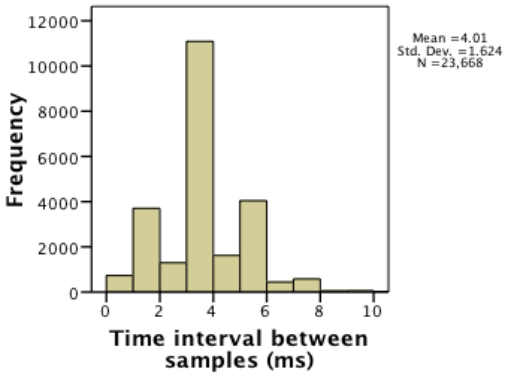

Figure 9: *Flutter in Vicon sample rate*

Conclusions and future work

Recording GDIF data to SDIF files as presented in this paper may solve many problems related to synchronization of data from different sources. In a recording situation, one would assume that analysis is meant to be done as post-processing, and so any preprocessing necessary for the analysis

could also be done subsequent to the recording. But, provided there is enough CPU power, it is possible to record higher GDIF layers already at this stage. The jmod.sdif.record module may also be used for recording higher level GDIF streams as part of the post-processing of data, and I believe it provides a good basis for further development of systems for storing and streaming GDIF data.

Advantages of using the methods presented here include the possibility to record any kind of numerical data from several sources independent of sample rate. It also provides a simple solution for streaming recorded data in raw or analysed format along with audio/video recordings of the performance. The disadvantages are the loss OSC coded information (since data needs to be recorded to fmat matrices). For streaming OSC data, the OSC namespace will need to be described in a name-value-table in the recording of the file, and re-applied during playback. Another disadvantage of using a real-time approach when recording data is the loss of possibilities for post processing. By recording data from the Vicon system to a 3rd party application, one loses many of the post-processing features that are available in the Vicon software. The Vicon system also allows streaming of audio and video simultaneously to the post-processing, so unless one wants to record data from other sensors, like the Polhemus or the T-stick, it is preferable to make recordings directly to the Vicon system.

Other approaches (like the MAV framework [Koerselman et al. 2007]) may also be evaluated for working with motion-related data. I have not focused on other approaches in this Short-Term Scientific Mission, and thus they have not been presented here.

The Jamoma SDIF modules will be improved by implementing a better solution for video synchronisation and support for standard SDIF types directly to the SDIF file as well as multi-stream playback from a single SDIF play module. Some graphical issues will also be improved.

The Jamoma modules presented here are available from the current active branch of Jamoma. Video examples, the patches in use in this presentation and some of the recorded files are available from http://folk.uio.no/krisny/research/ .

Acknowledgments

I would like to thank the Steering Committee of the COST-SID Action IC0601 for approving my Short-Term Scientific Mission. I would also like to thank the following people for help on the experiments and/or discussion and suggestions on the issues relevant to this Short-Term Scientific Mission: Joseph Malloch, Stephen Sinclair, David Savoca, Erika Donald, Fernando Rocha, Darryl Cameron, Alexander Refsum Jensenius and Marcelo Wanderley.

References

Jensenius, A. R., K. Nymoen & R. I. Godøy (2008): "A Multilayered GDIF-Based Setup for Studying Coarticulation in the Movements of Musicians" submitted for *the International Computer Music Conference 2008 (ICMC2008)*. Belfast, Northern Ireland

Jensenius, A.R. N. Castagné, A. Camurri, E. Maestre, J. Malloch and D. McGilvray (2007): "A Summary of Formats for Streaming and Storing Music-Related Movement and Gesture data" in *Proceedings of the 4th International Conference on Enactive Interfaces*, Grenoble, France.

Jensenius, A.R., A.Camurri, N.Castagné, E.Maestre, J.Malloch, D.McGilvray, D.Schwartz, M.Wright (2007): "PANEL: The Need of Formats for Streaming and Storing Music-Related Movement and Gesture Data" in *Proceedings of the 2007International Computer Music Conference*, Copenhagen, Dennmark.

Jensenius, Alexander R. (2007): *Action Sound: Developing Methods and Tools for Studying Music-Related Body Movement*, PhD thesis, University of Oslo. Oslo: Unipub

Koerselman, T., O. Larkin and K. Ng (2007): "The MAV Framework: Working with 3D Motion Data in Max/MSP/Jitter" in *Proceedings of the 3rd International Conference on Automated Production of Cross Media Content for Multi-channel Distribution*, Barcelona, Spain.

Malloch, Joe and Marcelo Wanderley (2007): "The T-Stick: From Musical Interface to Musical Instrument" in *Proceedings of the 2007 International Conference on New Interfaces for Musical Expression (NIME07)*, New York, USA. 66-69

Place, Tim and Trond Lossius (2006): "Jamoma: A modular standard for structuring patches in Max" in *Proceedings of the International Computer Music Conference 2006*, New Orleans, USA.

Sinclair, Stephen and Marcelo Wanderley (2007a): "Extending DIMPLE: a rigid body hapticsimulator for interactive control of sound" in *Proceedings of the 4th International Conference on Enactive Interfaces (ENACTIVE07)*, Grenoble, France. 263-266

Wright, M., A. Chaudhary, A. Freed, D. Wessel, X. Rodet, D. Virolle, R. Woehrmann & X. Serra. (1998): "New Applications of the Sound Description Interchange Format" in *Proceedings of the 1998 International Computer Music Conference (ICMC1998)*, Ann Arbor, Michigan

Wright, M., A. Chaudhary, A. Freed, S. Khoury, D. Wessel (1999): "Audio Applications of the Sound Description Interchange Format" in *Audio Engeneering Society 107th Convention #5032*, .