Motion tracking in musical instrument interfaces
SMA Technical Report.
Kristian Nymoen, November 2007
University of Oslo, Department of Musicology.

Abstract

In cognitive music research, ones main focus is the relationship between music and human beings. This involves emotions, moods, perception, expression, interaction with other people, interaction with musical instruments and other interfaces, among many other things. Due to the nature of music as a subjective experience, verbal utterances on these aspects tend to be coloured by the person who makes them. Such utterances are limited by the vocabulary of the person, and by the process of consciously transforming these inner feelings and experiences to words (Leman 2007: 5f). Thus, gesture research has become extensively popular among researchers wanting a deeper understanding of how people interact with music. In this kind of research, several different methods are used, using for example infrared-sensitive cameras (Wiesendanger et al. 2006) or video recordings in combination with MIDI (Jabusch 2006). This paper presents methods being used in a pilot study for the Sensing Music-related Actions project at the Department of Musicology and the Department of Informatics at the University of Oslo. Here I will discuss the methods for apprehending and analysing gestural data in this project, especially looking into use of sensors for measuring movement and tracking absolute position. In this project, a superior goal is to develop methods for studying gestures in musical performance. In a large view this involves gathering data, analysing the data and organizing the data in such a way that we ourselves and others easily can find and understand the data.

The sensors

We are working with sensors for measuring acceleration/movement, position and muscle contraction. For movement tracking we are using the *PhidgetAccelerometer 3-axis*. The accelerometer connect to a computer via USB, which makes it easy to work with on any computer. It outputs data at a sample rate up to 60 Hz, which should be more than enough for tracking human movements. The accelerometer outputs three values (X, Y, Z) referring to acceleration values along the three axis of a

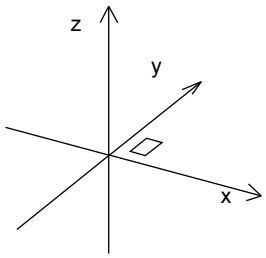

Figure 1: *Accelerometer in coordinate system*

three-dimensional coordinate system (figure 1).

The accelerometer outputs both dynamic acceleration, i.e. change in velocity per time unit, and static acceleration which means the current tilt of the accelerometer. The latter is added as a offset to the three values. The way this offset is added is not mentioned in the documentations of either the Phidget-Accelerometer 3-axis[1] or the ADXL330[2] (which is the acceleration chip on the accelerometer), but by looking at the data from the accelerometer, when it is kept still, it seems that this offset is added to the *x*, *y* and *z* axis as outlined in equation 1. This is shown by the accelerometer data in table 1.

$$\sqrt{x^2+y^2+z^2}=1$$

Equation 1: *relationship between accelerometer values*

Equation 1 shows that largest sum of offset added to the three variables is just below 1.8. This is relevant to the analysis at the end of this paper.

Example values from steady accelerometer			$\sqrt{x^2+y^2+z^2}$
x	***y***	***z***	
1,00	0,00	0,00	1,00
0,00	1,00	0,00	1,00
0,00	0,00	1,00	1,00
0,73	0,00	0,73	1,03
0,00	0,69	-0,69	0,98
0,00	0,71	0,71	1,00
0,00	-0,71	0,71	1,00
0,73	0,73	0,00	1,03
-0,71	-0,71	0,00	1,00
0,59	0,59	0,59	1,02
-0,57	0,57	-0,57	0,99
0,59	-0,59	0,59	1,02
-0,57	-0,57	-0,57	0,99
-0,79	-0,10	0,61	1,00
-0,17	-0,36	0,92	1,00
0,98	-0,29	0,06	1,02

Table 1: *Data from accelerometer lying still on table with different tilts. The data seem to follow equation 1. The divergence from 1 in the right column is probably due to me rounding off the accelerometer data in this table.*

The data from the accelerometer is measured in gravities (G), e.g. a data output of 1 means 9.81 m/s^2. According to the manufacturer, the accelerometer can sense acceleration of up to 3 G in each direction,[3] even so, it outputs data between -5 and +5. This is the sum of static and dynamic acceleration. I interpret this as the sensor being capable of measuring dynamic acceleration up to 4 G, but that when the dynamic acceleration exceeds 3 G, the data becomes less reliable.

1 http://www.phidgets.com/documentation/Phidgets/1059.pdf
2 http://www.analog.com/UploadedFiles/Data_Sheets/ADXL330.pdf
3 http://www.phidgets.com/documentation/Phidgets/1059.pdf

Accelerometer data			
Variable	**Useful Data Range**	**Physical Range**	
x	[-3 3] ± 1	[-3G 3G] ± 1G	Physical acceleration up to 3G, plus a "tilt offset" of ± 1G
y	[-3 3] ± 1	[-3G 3G] ± 1G	Physical acceleration up to 3G, plus a "tilt offset" of ± 1G
z	[-3 3] ± 1	[-3G 3G] ± 1G	Physical acceleration up to 3G, plus a "tilt offset" of ± 1G

Table 2: *Variables from the PhidgetAccelerometer 3-axis*

Polhemus Patriot data			
Variable	**Data Range**	**Physical Range**	
x	[0. 60.)	60 inches (≈ 152 cm)	Distance from the sensor to the *yz*-plane (absolute value)
y	(-60. 60.)	60 inches (≈ 152 cm)	Distance from the sensor to the *xz*-plane. The *y*-axis direction reverts when the sensor crosses the *yz*-plane
z	(-60. 60.)	60 inches (≈ 152 cm)	Distance from the sensor to the *xy*-plane. The *y*-axis direction reverts when the sensor crosses the *yz*-plane
azimuth	[-180. 180]	360 degrees	Twist (in degrees) around Z-axis of the sensor. Value 0 when the sensors x-axis aligns with the absolute x-axis
elevation	(-90. 90.)	180 degrees	Angle (in degrees) between the X-axis (along the cable) of the sensor and the absolute XY-plane
roll	[-180. 180]	360 degrees	Twist (in degrees) around the X-axis of the sensor. Value 0 when the sensors z-axis aligns with the absolute z-axis

Table 3: *Variables from a Polhemus Patriot sensor*

We are using a *Polhemus Patriot*, electromagnetic tracking system to determine absolute position in the room. This system has two sensors moving in a electromagnetic field (EMF). The device is able to determine the absolute positions of the sensors by less than 1 mm accuracy, within a range of about 1.5 m from the source of the EMF (Maestre *et al*. 2007). In addition the sensor outputs data on how it is tilted (*azimuth*, *elevation*, *roll*).

The *x*, *y* and *z* coordinates are in units of inches. The *x*-coordinates are absolute values (i.e. all coordinates are output as values larger than 0). The *y*- and *z*-coordinates are scaled along the *y*- and *z*-axis and are output as both positive and negative values along the axis. These axis change direction when the sensor crosses the *yz*-plane (see figure 2).

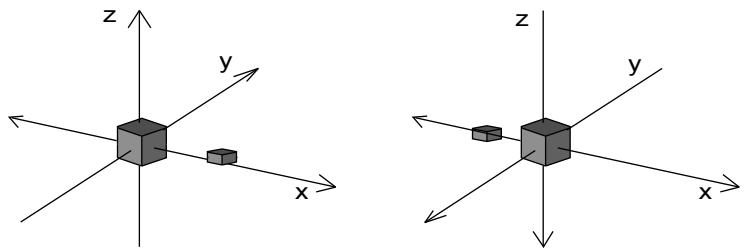

Figure 2: *Axis orientation, according to sensor position. x-coordinates are always output as positive values. The y- and z-axis change direction when the sensor crosses the yz-plane. The large cube in the centre is the source of the EMF, and the smaller cube is the sensor.*

We are also using *I-cube X electromyography biosensors* (EMG) which measures muscle contractions. These have not been my main field of study, so I mention them here only briefly as I will refer to some of these data below.

The data

We are using *Max/MSP*[4] for all communication with the sensors and recording of data. Some extensions to this software has been particularly important from a methodological view and will be outlined here.

FTM[5] has been developed at IRCAM as mainly as an extension to Max/MSP. In our use, the handling of the *Sound Description Interchange Format* (SDIF) in FTM is the most important feature. SDIF was developed at IRCAM as a standard for storing and exchanging sound data (Schnell *et al*. 2005). The SDIF format consists of separate streams along a common timeline. Each of this streams is a sequence of time-tagged frames. Within the frames, the actual data are stored into matrices. The frames and the matrices are type-specific, meaning that different frame types consists of a different amount of matrices, and different matrix types consists of a different number of rows and columns (Wright *et al.* 1998). The structures of matrices and timelines in FTM is well suited for recording different types of data along a common timeline, an so it is also well suited for recording of SDIF files.

The *Gesture Description Interchange Format* (GDIF), inspired by SDIF, is currently under development as a standard for communicating (i.e. storing and streaming) gesture related data (Jensenius *et al*. 2006). GDIF is a multiple-layer format consisting of quantitative *raw data* from the sensors, *pre-processed data* and *analysed data*. In addition to the quantitative data, GDIF also includes qualitative metadata, annotations and observations (ibid). As GDIF development is mainly focused on what to store, rather than how to store it, recordings can be made in different types of formats, such as XML or SDIF (Jensenius *et al*. 2007b). The distinction between pre-processed data and analysed data seems somewhat unclear as Marshall *et al*. (2006) only separates between *raw data* and *body data* as quantitative values, and neither Jensenius *et al*. (2006) or Jensenius *et al*. (2007a) outlines whether the pre-processing of data only involves scaling the data to e.g. a [0.–1.] range, or if it involves scaling the sensor values to a common reference point for multiple sensors. In my understanding, pre-processing data only involves scaling the raw data in itself, meaning that scaling the data to a common reference point would be part of the analysis rather than pre-processing.

When recording data to GDIF, a first challenge is to organize and process the raw data. Jensenius *et al*. (2006) suggest using *Open Sound Control* (OSC) to organize the data in the GDIF. This way the raw data from the sensors can easily be separated from the pre-processed and analysed data. Marshall *et al*. (2006) outlines the importance of a consensus on a namespace for these data.

4 http://www.cycling74.com
5 http://ftm.ircam.fr/

They suggest storing raw data in the format:

/device/sensor <*data*>

which means that for a Polhemus Patriot, we would get two streams:

/patriot1/sensor1 *x y z azimuth elevation roll*

/patriot1/sensor2 *x y z azimuth elevation roll*

A namespace for analysed data, relating to the body parts is suggested as:

/body/part/side <*data*>

which could mean:

/body1/arm/left *x y z azimuth elevation roll*

In my opinion, a universal predefined namespace is somewhat problematic, as different projects using different sensors placed on different parts of the body would mean that a very large amount of redundant data is necessary to cover all different types of setups. For example, one project may aim to use a Polhemus sensor to describe the positioning of the left and right arms of different subjects. In this example the namespace from the example above is sufficient. Another example would be using three Polhemus Liberty sensors for measuring the position of the wrist, elbow and shoulder on the left arm of a single subject. Since we are using only one person, and only the left arm, both /body and /side would be redundant messages being stored for every single data input, while /part would change meaning, referring to a body part on a lower level (e.g. *wrist* as opposed to *arm*). Due to the large amounts of redundant data that would have to be stored into every GDIF-file in this case, I see it as necessary to define a namespace for each GDIF-file. To do this, the header of a GDIF-file should contain information on the sensors being used (the sensor types and the number of sensors) and the data from each sensor (number of variables, data type and data range). Further on, the header should contain information on the placement of the sensors on the subject, making it possible to see which raw data matches which analysed data.

Jensenius *et al.* (2006) suggests a higher level stream, where metadata (i.e. qualitative descriptions of the gestures) are stored. Marshall *et al.* (2006) organizes these types of data in a separate /meta stream.

In my second example above, one may want to record the raw data, and adjust the Polhemus data to be relative coordinates to the shoulder coordinates (to keep things simple, I only use the Cartesian coordinates in this example). One could additionally want to add a pre-processed layer, for scaling the values to another length unit (e.g. millimetres), I choose not to do this in this example. In addition to calculating relative coordinates, it would be relevant to look at the speed and acceleration of both the wrist, the elbow and the shoulder, and to make a quantitative description of the contraction and the elevation of the arm. These parameters are shown in table 4.

Example SDIF parameters	
Raw data	*x y z azimuth elevation roll (×3)*
Processed data	x_{rel} y_{rel} z_{rel} $azimuth_{rel}$ $elevation_{rel}$ $roll_{rel}$ *(×3)* *Contraction* *Elevation* *Speed* *Acceleration*

Table 4: *Parameters in use in example, the suffix $_{rel}$ are parameters adjusted relative parameters. (×3) means data being stored for each sensor*

The parameters *contraction, elevation*, *speed* and *acceleration* does not refer to a single sensor parameter, but are results of calculations based on several parameters. They introduce a possible need for expanding the quantitative part of the namespace beyond the raw, pre-processed and analysed data. These parameters does not only describe the same type of data as the sensor (in this example cartesian coordinates + tilting angles), but use the sensor values to achieve other types of information. In this example, one may benefit from an OSC namespace where the analysis part of the namespace is divided into /relative and /analysis. The namespace is displayed in table 5.

Raw data:

/raw/liberty1/sensor1	x_1	y_1 z_1	$azimuth_1$	$elevation_1$	$roll_1$
/raw/liberty1/sensor2	x_2	y_2 z_2	$azimuth_2$	$elevation_2$	$roll_2$
/raw/liberty1/sensor3	x_3	y_3 z_3	$azimuth_3$	$elevation_3$	$roll_3$

Relative (to shoulder) sensor data

/relative/arm/wrist	x_w	y_w	z_w
/relative/arm/elbow	x_e	y_e	z_e
/relative/arm/shoulder	0	0	0

Analysed data

/analysed/arm	*contraction*	*elevation*		
/analysed/arm/wrist	*abs_speed*	*abs_acceleration*	*rel_speed*	*rel_acceleration*
/analysed/arm/elbow	*abs_speed*	*abs_acceleration*	*rel_speed*	*rel_acceleration*
/analysed/arm/shoulder	*abs_speed*	*abs_acceleration*	*rel_speed*	*rel_acceleration*

Metadata

/meta/expressivity	*value*
/meta/randomness	*value*

Table 5: *OSC namespace for the SDIF example*

Here, the relative data in /wrist, /elbow and /shoulder are displaced to values based on the raw data with the shoulder sensor as a reference point, meaning that the shoulder coordinates are (0, 0, 0,). The coordinates (x_w, y_w, z_w) would then be equal to (x_1–x_3, y_1–y_3, z_1–z_3) and (x_e, y_e, z_e) would be found by (x_2–x_3, y_2–y_3, z_2–z_3). *Contraction* could be estimated in several ways, the easiest way probably to look at the point (x_w, y_w, z_w). The magnitude of the vector from this point to the reference point would be a value reflecting the contraction of the arm (the larger the value, the more stretched out the arm is). Speed and acceleration are calculated by derivatives of vector magnitudes between succeeding

room coordinates. Both absolute and relative speed and acceleration could be of interest, and are thus included in the example above.

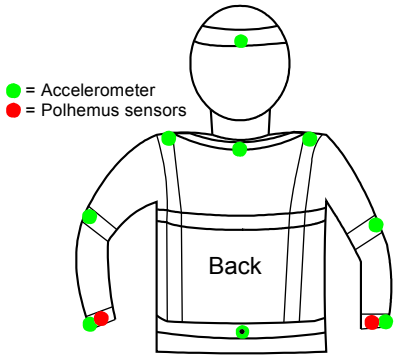

Figure 3: *Positions of the accelerometers and polhemus sensors*

The practical example

In our project so far, we have made a limited number of recordings of gesture data from a piano player. We used nine accelerometers to track movement in different parts of the body. Four of these were mounted on an adjustable "strap-on" system, to to be strapped on the pianists upper body: One accelerometer on each shoulder, one on the back of the neck, and one accelerometer on the lower back. Additionally, one accelerometer was strapped on the back of the pianist's head, and two accelerometers on each arm; one close to the elbow and one close to the wrist. The two Polhemus Patriot sensors were mounted next to the accelerometers on the wrist (see fig. 3). For a small part of these first recordings we used two EMG sensors to measure muscle contractions on the lower arm; however, the sensor on the right arm did not work during the pilot recordings. In this paper I refer

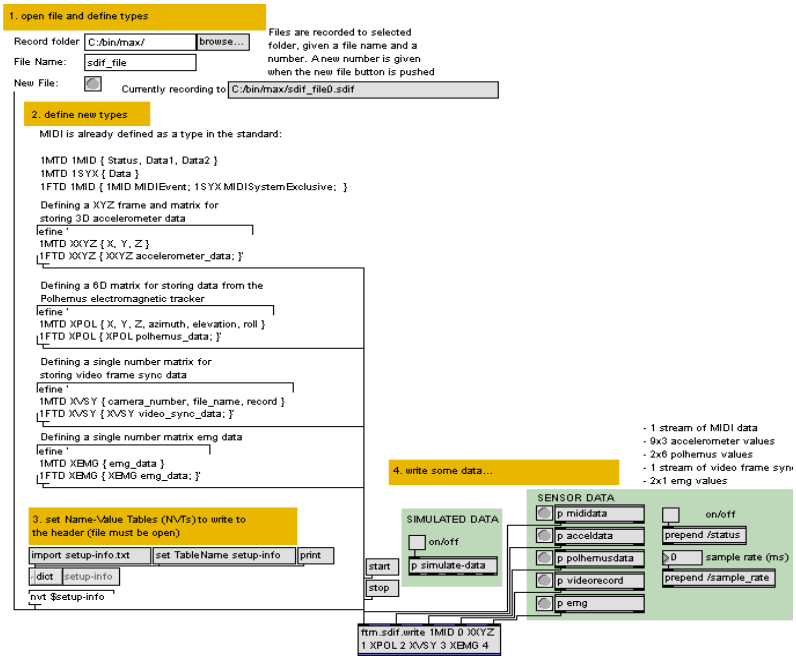

Figure 4: *Max/MSP patcher being used for recording SDIF-files. Main setup, including FTD and MTD definitions made by Alexander Refsum Jensenius.*

less to the EMG data than to the accelerometer and Polhemus data. In addition to the sensors, we recorded MIDI data from the piano being played on (Yamaha disklavier).

For our project we are recording to SDIF-files, recording data from each type of sensor into a separate SDIF-stream. This was done by defining separate matrix types and frame types for the different data types. As we are planning on making video recordings on a separate computer we also needed a synchronisation frame and matrix. In our project, we used a complete setup within in Max/MSP. Max/MSP communicates with all the sensors we are using, and within the FTM structures it is easy to make well-organized SDIF recordings. When analysing the data, FTM allows isolating only the streams one wants to look into, by importing only the wanted streams into FTM tracks. When making SDIF recordings with the ftm.sdif.write object by IRCAM, every frame is automatically given a time tag which is helpful when comparing data from different streams and analysing the development of sensor data over time. I will look into the process of analysing recorded data, with examples from our pilot recordings in the next section.

The analysis

Analysing raw data may involve both ascribing a significance to each of the sensor values alone, and looking at coherences in a selection of raw data. In this discussion, I will mainly look into problems with and analysis of data from the Polhemus Patriot and the accelerometers. I do not address the video sync stream, as we have not recorded data into this stream yet. Table 6 presents a simplified graphical view of a cross section of the data in our SDIF recordings. The file consists of five streams, where each stream consists of succeeding time-tagged frames, and every frame consists one matrix of data. Every frame and every matrix has its own unique ID, making it easy to look up data. As the frames are time-tagged, one can look at timing differences between e.g. the initial movements

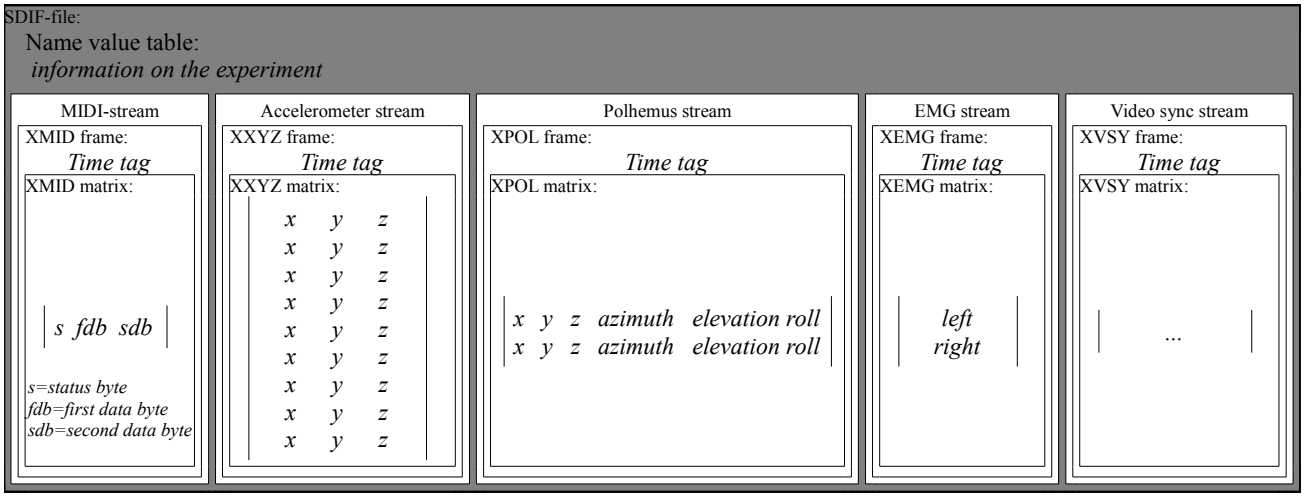

Table 6: *A simplified graphical view of a cross section of our SDIF-files. The different streams consists of different types of frames and matrices (XMID, XXYZ, etc.). The user can easily navigate to a specific time, or isolate a single stream e.g. to only look at data from a single sensor.*

in the accelerometers and the keys being struck on the piano.

In our pilot recordings, I have found some problems in the Polhemus setup which complicates the analysing of the sensor data. The source of the electromagnetic field was placed in such a way that the pianist was facing almost along the *x*-axis, the keyboard was almost aligned along the keyboard, and the *z*-axis was vertical. As mentioned earlier, the *z*- and *y*-axis changes polarity when

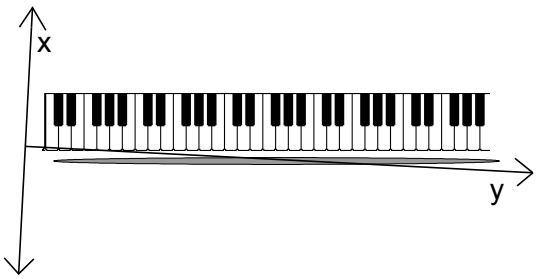

Figure 5: *Sketch of Polhemus x/y-axis. The grey area illustrates the area where the sensors mainly were moving. The z axis is along the reader's view of the figure.*

the sensor crosses the *zy*-plane. As illustrated in figure 5, the sensors were moving on both sides of the *zy*-plane, causing *z*- and *y*-values to flip between positive and negative values. This flip, however, does not seem to be immediate, but rather an interpolation during a time of about 50 ms between the positive and negative equivalents of the value, giving smaller, false values in the samples between the two extremities. As an example, six succeeding *y*-values from the right hand of the player are presented in table 7. This means that an absolute value, which would have solved the

-29.0	-29.1	11.2	25.0	28.3	29.1

Table 7: *Six succeeding y-values from the Polhemus sensor on the right wrist. The time between each of these values is 17 ms, meaning that if one would assume absolute values of these data to be correct, the pianist would have moved her hand appropriately 18 inches (46 cm) in 17 ms and almost all the way back within the next 17 ms.*

problem if it was a direct flip with no interpolation, is insufficient. To make the errors as small as possible, I have found it useful to filter out all the smallest *y*-values for both sensors. To determine the threshold for which values that should be included in the pre-processed data, I counted all the instances of *y*-values between 0 and 1, 1 and 2, and so on up values between 39 and 40 in one SDIF-file. This was done in a Max-patcher using select and counter objects. For the left hand there was a great difference between the intervals 7-8 (0 instances) and 8-9 (135 instances), while for the right hand, the greatest difference was between the intervals 21-22 (53 instances) and 22-23 (353 instances). This means that 8 and 22 are suited thresholds for the right and left hand respectively. Figure 6 shows a part of this patcher, displaying the interval counts in a multislider for each sensor. There is reason to believe that the error margins are larger on the right hand, as this was both the sensor furthest away from the EMF source, and the sensor crossing the *y*-axis most times. For this reason I have chosen to interpret the small "tail" in the area between 12 and 22 on the right hand (see figure 6) as errors.

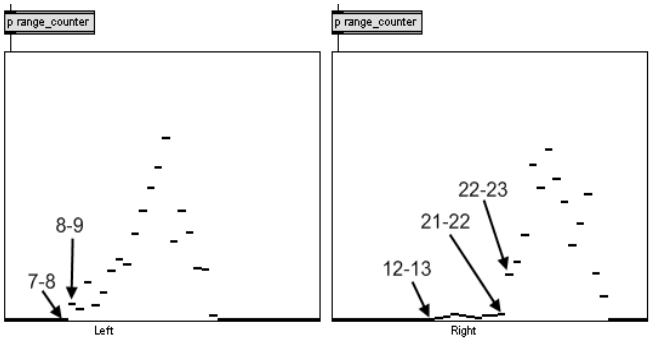

Figure 6: *Part of patcher for counting occurrences of the y-coordinate within different intervals.*

The next problem with the Polhemus data is that *x*-values are only positive values. Looking back at figure 5 it is obvious that the *x*-values should be interpreted as both positive and negative values, to get the correct room coordinates. This easiest solution to this problem is a simplification by setting a single *y*-coordinate where *x*-values turn from negative to positive. To find this coordinate, one could use linear regression on all (*x*, *y*) points with *y*-values within a certain interval. Unfortunately, I do not have the tools for doing linear regression, so my already inaccurate method becomes more inaccurate when I estimate the crossing point by looking at the table object in figure 7. The *x*-values seem to approach 0 when the *y*-values approach 25. Again referring to the sketch in figure 5, I choose to set *x*-values as positive when $y > 25$ and as negative values when $y < 25$. There are similar problems related to the *z*-values from the Polhemus.

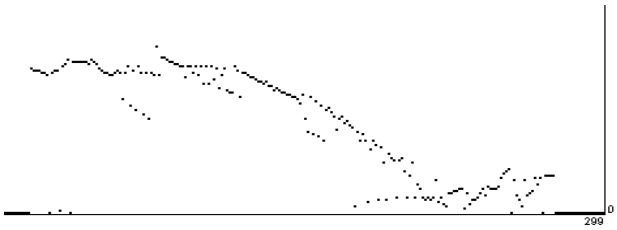

Figure 7: *Table object in patcher for looking at where the x-values change polarity.*

When seeking information about movement, the static acceleration offset added to the accelerometer values is not as relevant as the dynamic acceleration. This can be filtered out with a high pass filter. After some trial-and-error of different filters I ended up in this example with a simple dual-stage FIR-filter with coefficients of 0.5 and -0.5.[6] Since each of the accelerometers outputs three values, and in this example, we are only interested in the total acceleration for each sensor, I calculated the sum of the absolute values of the three variables. This number was then low-pass filtered, to remove high-frequent noise. This setup is shown in fig. 8, and the result of the analysis of three accelerometers on the right arm is shown in fig. 9. The unfiltered data (left) is hard to inter-

6 FIR = *Finite Impulse Response*, coefficients of 0.5 and -0.5 means that every output sample is a result of the current input sample multiplied by 0.5 + the previous input sample multiplied by -0.5 (Roads 1996: 413).

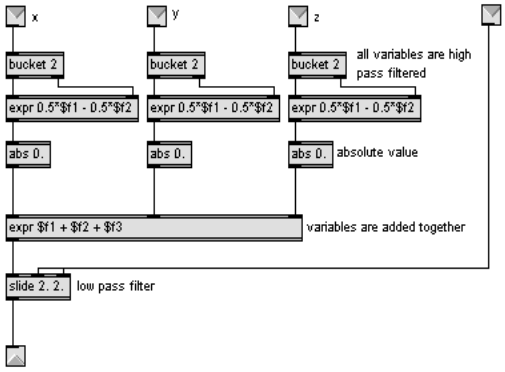

Figure 8: *Filtering of the accelerometer values in Max/MSP*

pret, as there is a lot of noise, and the unnecessary offset is making it harder to read. In the filtered data (right) it is easy to extract the acceleration peaks, and thereby to compare the data from the three accelerometers.

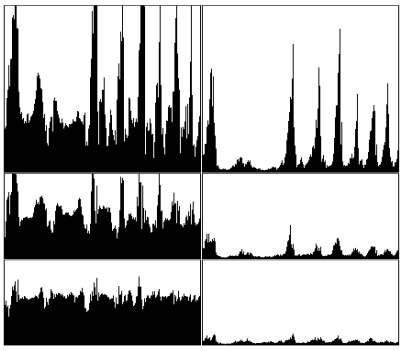

Figure 9: *Display of unfiltered (left) and filtered (right) accelerometer values. Top: wrist, middle: elbow , bottom: shoulder*

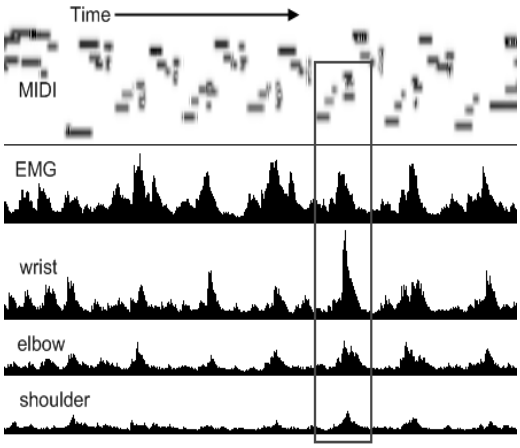

Figure 10: *Plot of accelerometer values from the left wrist, elbow and shoulder, and EMG from the right lower arm. Each horizontal pixel represents a time of 17 ms.*

Since we have recorded MIDI data, it is possible to compare the MIDI velocities to the various types of sensor data. The velocity data is closely tied to loudness and sound energy, and thus it would be interesting to see if there is some correlation between the velocity and the acceleration values or the EMG.

To study the relationship in starting times for each of the accelerometer peaks, the midi note-on and note-off messages and the EMG, it would be useful to plot these data along a common timeline. This has been done in figure 10. Looking at the plots inside the square, we can see that the lower arm EMG begins its peak slightly before the wrist accelerometer. This again is slightly before the right hand cluster (the three notes played on the same time, displayed on the MIDI plot inside the square) which is emphasised (the dots are darker than the preceding ones, meaning the midi velocity is higher). To draw significance from these comparisons, one will need to study the data closer, looking at exact time values and having meaningful and comparable values on the vertical axis (in figure 10 the accelerometer parameters have been filtered like in figure 9, but the vertical axis have been

scaled individually for better resolution, making the plots less comparable). A better analysis would be achieved by including a video recording of the performance. It would then be possible to extract the frames where the accelerometers rapidly change value due to tilting. As stated before the maximum offset added as a total to the three variables is 1.8, meaning that before the low-pass filtering an error of 1.8 would be recorded if the subject was to tilt the accelerometer from one extreme to the other within 17 ms (i.e. the sampling rate). This is quite unlikely to happen, but it would be a significant error, as the wrist accelerometer in figure 10 varies between 0 and 3.4.

Conclusions

In the field of gathering and structuring gesture data, the biggest challenge at the moment is probably to standardize the communication of this data. The GDIF format is a good starting point for this, but in my understanding it is still different practices on what to include in the GDIF, and on the namespace for the included data, and it does not seem to be a common opinion on how data should be processed. The best way of standardizing all of this is probably to keep using the format and document the advantages and disadvantages of the setup in each project, allowing future projects to reach a common structure that is both stable and that covers all the needs in storing gesture-related data.

In this paper I have not addressed the problems that occurs when one places sensors on a musician. It is more than likely that an excessive amount of cables and extra weight does affect the performance. Such problems can not easily be solved when working on a tight budget, but may be solved by more advanced motion capture systems like Vicon (O'Sullivan/Igoe 2004: 228). I have outlined some problems in our first recordings, related to the sensors themselves. These will be corrected, and hopefully data from our next recordings will be easier to work with. Such testing recordings and investigation of the test data seem to be an important part of the preparing the data for analysis. From my examples, it is obvious that a small change in sensor placement (in my case rotation of the EMF source) can make less pre-processing of data necessary, which would mean less loss of information before the analysis.

References

Jabusch, H. C. 2006 “Movement analysis in pianists” in E. Altenmüller, M. Wiesendanger and J. Kesselring. Eds. 2006: *Music, Motor Control, and the Brain*. Oxford University Press. New York.

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 (ed.): *Proceedings of New Interfaces for Musical Expression, NIME 06, IRCAM - Centre Pompidou, Paris, France, June 4-8*, pp. 176–179. Paris: IRCAM - Centre Pompidou.

Jensenius, A. R., A. Camurri, N. Castagne, E. Maestre, J. Malloch, D. McGilvray, D. Schwarz and M. Wright. 2007a. “Panel: the Need of Formats for Streaming and Storing Music-Related Movement and Gesture Data” in *Proceedings of the 2007 International Computer Music Conference, Copenhagen*.

Jensenius, A. R., N. Castagné, A. Camurri, E. Maestre, J. Malloch and D. McGilvray. 2007b: “A Summary of Formats for Streaming and Storing Music-Related Movement and Gesture data” in *Proceedings of the 4th International Conference on Enactive Interfaces (Enactive07)*, Grenoble, France

Leman, M. 2007: *Embodied Music Cognition and Mediation Technology*. MIT Press

Maestre, E., J.Bonada, M. Blaauw, A. Perez and E. Guaus. 2007: “Acquisition of violin instrumental gestures using a commercial EMF tracking device” in *Proceedings of the 2007 International Computer Music Conference (ICMC2007)*. Copenhagen, Denmark.

Marshall, M.T., N. Peters, A.R. Jensenius, J. Boissinot, M.M. Wanderley and J. Braasch. 2006: “On the Development for Gesture Control of Spatialisation” in *Proceedings of the 2006 International Computer Music Conference, 6-11 November, New Orleans*.

O'Sullivan, Dan and Tom Igoe. 2004: *Physical computing : sensing and controlling the physical world with computers (E-book version)*. Boston: Thompson.

Roads, Curtis. 1996: *The Computer Music Tutorial*. Cambridge, Mass.: MIT Press.

Schnell, N., R. Borghesi, D. Schwarz, F. Bevilacqua, R. Müller. 2005: “FTM — Complex data structures for Max” in *Proceedings of the 2005 International Computer Music Conference (ICMC2005)*, Barcelona, Spain

Wiesendanger, M., A. Baader, and O. Kazennikov. 2006: “Fingering and bowing in violinists: a motor control approach” in E. Altenmüller, M. Wiesendanger and J. Kesselring. Eds. 2006:

Music, Motor Control, and the Brain. Oxford University Press. New York.

Wright, M, A. Chaudhary, A. Freed, D. Wessel, X. Rodet, D. Virolle, R. Woehrmann and X. Serra. 1998: "New applications of the Sound Description Interchange Format" in *Proceedings of the 1998 International Computer Music Conference*, Ann Arbor, Michigan,
URL: http://www.cnmat.berkeley.edu/ICMC98/papers-html/SDIF.html [November 19, 2007]

Other

Analog devices, ADXL330 manual.
URL: http://www.analog.com/UploadedFiles/Data_Sheets/ADXL330.pdf [November 1, 2007]

Cycling'74: Max/MSP.
URL: http://www.cycling74.com

FTM, IRCAM
URL: http://ftm.ircam.fr/

PhidgetAccelerometer 3-axis manual.
URL: http://www.phidgets.com/documentation/Phidgets/1059.pdf [October 31, 2007]

Polhemus Patriot manual.
URL: http://www.polhemus.com/polhemus_editor/assets/PATRIOT%20Manual.pdf [November 13, 2007]