”Extension du Corps Sonore ” - Dancing Viola

Todor Todoroff
ARTeM and FPMs TCTS
273, Rue du Progrès
1030 Bruxelles BELGIUM
31, Bd. Dolez
7000 Mons BELGIUM
todor.todoroff@skynet.be

Frédéric Bettens
Loïc Reboursière
FPMs TCTS
31, Bd. Dolez
7000 Mons BELGIUM
frederic.bettens@fpms.ac.be
loic.reboursiere@fpms.ac.be

Wen-Yang Chu
UCL TELE
2, Place du Levant
1348 LLN BELGIUM
wen-yang.chu@uclouvain.be

Abstract

”Extension du corps sonore” is long-term project initiated by Musiques Nouvelles [4], a contemporary music ensemble in Mons. It aims at giving instrumental music performers an extended control over the sound of their instrument by extending the understanding of the sound body from the instrument only to the combination of the instrument and the whole body of the performer. The development started at ARTeM and got the benefit of a three month numediart research project [1] that focused on three axes of research: pre-processing of sensor data, gesture recognition and mapping through interpolation. The objectives were the development of computing methods and flexible Max/MSP externals to be later integrated in the ARTeM software framework for the concerts with viola player Dominica Eyckmans. They could be used in a variety of other artistic works and will be made available on the numediart website [1], where more detailed information can be found in the *Quarterly Progress Scientific Report #4*.

Keywords: Sensor data pre-processing, gesture recognition, mapping, interpolation, extension du corps sonore

1. Introduction

Contrarily to the usual augmented instruments designs that track the gestures used to play the instrument to expand its possibilities, this project focuses on using non-musical gestures to transform the sound of the instrument. Our approach is dictated by the nature of Dominica’s project: she is actually dancing while playing the viola and we track her dancing movements rather than her hands movements. Research and experiments with Dominica Eyckmans started in the beginning of 2008 at ARTeM, using a Wi-Fi sensor system developed in 2006 for violist Stevie Wishart for the *Quartet* project [2]. The software framework, developed inside the Max/MSP environment to map sensor data to parameters of various sound transformation algorithms, has steadily evolved since the *Quartet* project, mostly thanks to the work on the dance performance *De deux points de vue* [3] choreographed by Michèle Noiret and premiered in December 2007. These two previous projects allowed to identify the specific needs related to both instrumental and dance performances. And though we found many ways to use the tools reliably and creatively, it also demonstrated the interest of deeper research in order to overcome some limitations. We wanted to better extract features directly from the sensor data, like hit detection, with very low latency. And we also wanted the benefit of more complex filtering techniques to obtain additional mapping inputs from the same sensors. Though inherently less responsive than mapping, gesture recognition is a welcome addition to our interactive performance and can be used to trigger events, to adapt the response of the virtual instruments according to the detected gestures or to move through the various steps of a performance. The nature of our project makes it difficult for the performer to notify the start and end of a movement as dance gestures are usually chained without pauses. As we also wanted to add more global ways of mapping parameters while keeping the responsiveness of the system, we decided to improve the interpolation scheme, inspired by [5], that we had developed previously on the NeXT platform [6]. This

Figure 1. Dominica Eyckmans, dancing and playing the viola

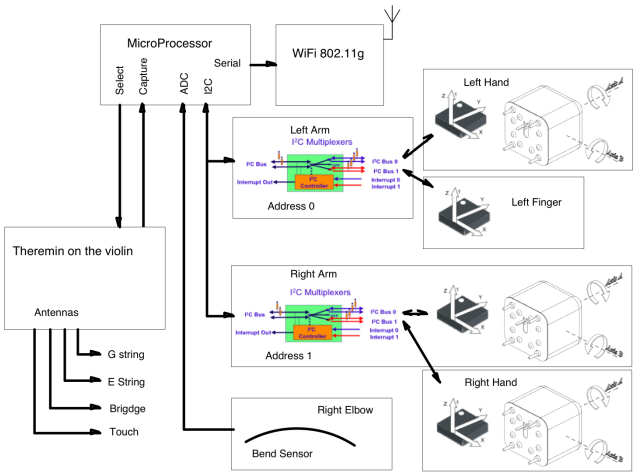

Figure 2. Sensors system developed for Quartet

report is divided in four main sections: after a brief review of previous art related works, we present a global view of the system, followed by the `numediart` developments on pre-processing of sensor data in order to extract features with minimum latency, on gesture recognition without the need for segmentation and on an improved mapping tool based on interpolation.

2. System

2.1. Sensors

The sensors system developed at ARTeM for the *Quartet* [2] project allowed for the data of a variety of sensors (theremin antennas, flexion, I2C accelerometers and I2C gyroscopes), placed on the violin and on the body of the performer to be transmitted every 8ms wirelessly over Wi-Fi (Figure 2). For this project we decided to downsize the sensor system and explore how far we could reach with sensors placed only on the ankles, each a combination of a 3-axis accelerometer and a 2-axis gyroscope. A similar setup was used for the dance performance *De deux points de vue* [3] , except that the sensors where placed on the upper arms of the dancer. The placement on the ankles presents a minimal hinderance even for movements on the ground. Depending on the results of further experimentation with the new software tools, we will consider the need, the type and the placement of additional sensors on Dominica's body.

2.2. Software framework

The ARTeM software is organized around a modular concept: the audio paths of the various virtual instruments are connected through a matrix, with external inputs and outputs of virtual instruments injected from the top and redirected with selectable level, thanks to a control object developed by Nick Rothwell, to the inputs of the virtual instruments and the external sound outputs on the right (Figure 3).

The sensors data are received as UDP packets through the integrated Wi-Fi interface. An Max external decodes the custom protocol, scales the raw data, and defines a name

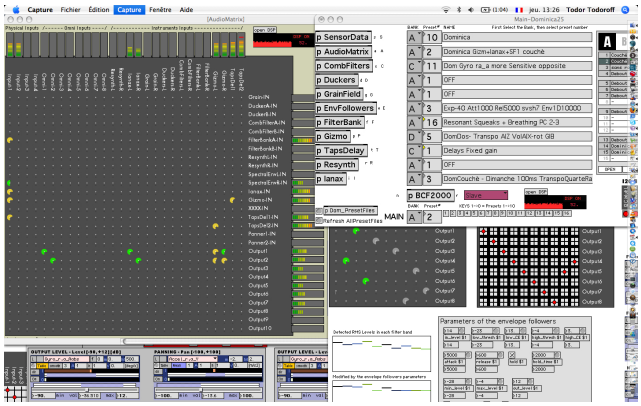

Figure 3. The audio matrix and some virtual instruments

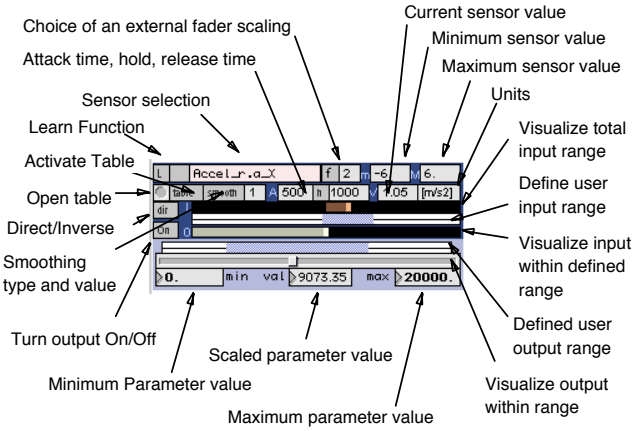

Figure 4. Mapping associated to each parameter

space depending on configuration messages sent to its input. It then outputs data as messages, available through a send/receive scheme throughout all the patches.

The scaling of the parameters is quite versatile and happens through a Max *bpatcher* (Figure 4) that is used in every place in the virtual instruments where parameters need to be given values from the sensors. It offers various stages:

- when the desired sensor is selected from a menu, the minimum, maximum and current values, and the units, here m/s^2 for an accelerometer, are displayed.
- a portion of the whole range of the sensor data can be defined using the Max *rslider* GUI object. The *learn* function can be activated to define the range that comprises all the values reached by a specific sensor during a particular set of movements.
- the values can then be transformed by a table that can be opened and drawn by the user.
- the value can also be reversed and turned on or off.
- a smooth menu offers the choice between a min, mean, median or max function followed by the number of data points, or a slide function.
- a simple attack, hold and release envelope generator is activated whenever one of the times are above $1ms$.
- this output is then mapped to the entire excursion of a slider that continuously displays the values.

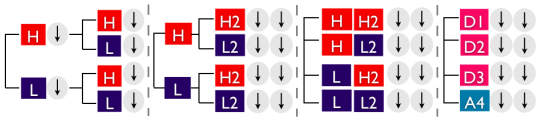

Figure 5. From classical wavelet packets to undecimated wavelet packet decomposition (UWPD)

- an output range selector allows to map those values within a range corresponding to the minimum and maximum acceptable range of the sound parameter. In the figure above, 0 to 20000 correspond to min and max center frequency of a filter. And those values can also be changed within those limits to allow for a very precise control of the parameter value.
- a fader or rotary button value can be inserted in the previous stage to allow the sound engineer to tune the scaling live using normal or motorized MIDI faders.

Audio analysis tools, like envelope followers on the incoming audio signals or at the output of filter banks, or *iana~* [9], generate data that can be used exactly in the same way as sensor data through all the patches. It allows to use the tools we describe hereunder undiscriminately on sound analysis data, on sensor data, or on combinations of both.

3. Preprocessing

We wanted to detect hits with accelerometers attached on the limbs of a performer, and extract scale, direction, and other possible frequency features with a minimum latency. A first hit detector was implemented, with parallel median filters with different window sizes to give an indication about the speed of the hits.

To get more frequency information, we propose an undecimated wavelet packet method. We chose to use wavelets [14] because of their better temporal resolution for the higher frequency content compared to FFT. We decided to use the Haar wavelet as it is the best candidate to detect transitions because of its short impulse response. In order to be able to reconstruct the signal continuously instead of frame per frame, for filtering purposes, we choose the wavelet packet approach (Figure 5a) [15]. To avoid the decimations, which reduce the data rate, we permute the order between filters and decimations in Figure 5b by upsampling the H and L filter to become $H2$ and $L2$. We then further combine the filters in Figure 5c to obtain the equivalent wavelet filter bank $D1$, $D2$, $D3$, $A4$, shown in Figure 5d. Finally we obtain the undecimated wavelet packet decomposition(UWPD) by discarding the accumulated decimations, getting a single filter for each channel, with the same output data rate.

With coefficients obtained through those filters, an algorithm detects whether a hit appears, in which axis, its direction, scale, and center frequency (Figure 6). As the energy of the hits usually spreads over several axes, the algorithm has to determine the presence of a hit on the basis of the energy

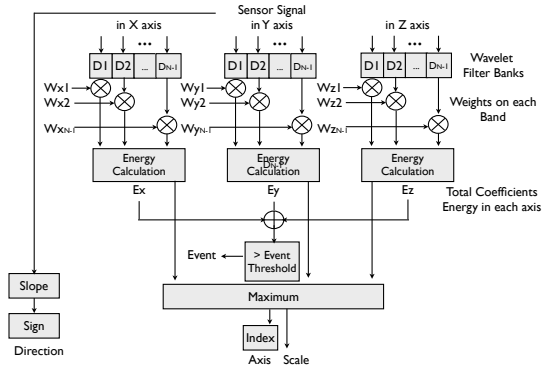

Figure 6. System flowchart of the UWPD filter banks method

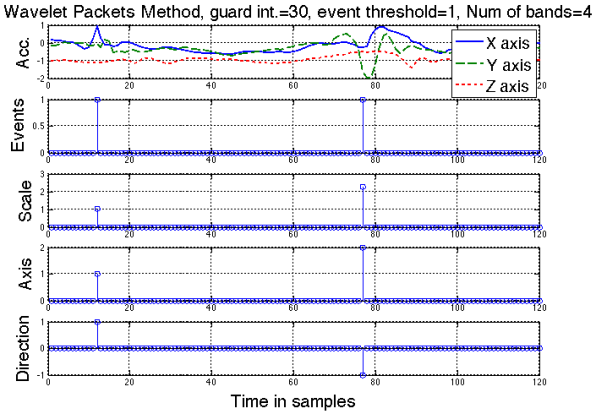

Figure 7. Detection results of the Wavelet packets method with threshold=1

of all the coefficients on all three axes. A matrix of weights, composed of W_{Xn}, W_{Yn}, W_{Zn} $(n = 1, 2, ..., N-1)$, allows us to control the sensitivity of each axis in each frequency band n. We may then define preferential directions as well as target specific frequency content. E_X, E_Y, and E_Z are used to detect the direction of a hit biased towards the targeted direction and desired frequency response. With unity weights, the detection results with a guard interval of 30 are shown in Figure 7.

4. Gesture recognition

In the context of the Dancing Viola project, we would like to be able to recognize a set of pre-defined gestures on the basis of the temporal evolution of accelerometer and gyroscope data [11]. As in Automatic Speech Recognition (ASR) [10], we are faced to the issue of comparing different realizations that can vary in duration while representing the same (gestural/vocal) intention. Whereas statistical (HMM-based) methods require a training stage, we prefer the DTW approach, as it is desirable that some gestures could be added, removed, enabled, or disabled easily and quickly. Also, this solution is suitable when developing a user-dependent recognition system working with a small amount of reference gestures (cf. small vocabulary in ASR).

4.1. The DTW algorithm

The DTW algorithm searches for the best non-linear mapping between the temporal indices of the test sequence ($i = 1..I$) and those of the reference sequence ($j = 1..J$). We will denote by $d(i, j)$ the "local distance" (or dissimilarity measure) between the test frame i and the reference frame j (where a frame is composed of the data of all sensors and axes at a given instant), and by $D(i, j)$ the "accumulated distance" along the sub-path between the origin $(1, 1)$ and current node (i, j), computed by a weighted sum of the local distances:

$$D(i,j) = \sum_k W(i_k, j_k; i_{k-1}, j_{k-1})\, d(i_k, j_k) \qquad (1)$$

These transition costs W raise the question of normalization, when comparing paths of different lengths (in the same DTW grid), or when comparing a given test gesture to several reference gestures of different durations. The normalization factor is path-independent $(I + J)$ when using symmetric transition costs [13].

4.1.1. DTW search constraints

We use the global path Itakura constraints where maximum allowable compression and expansion factors (λ_{max} and λ_{min}) can be defined by the user.

Local path constraints are defined by indicating the sets of predecessors to a given node [12]. In other words, the legal transitions are listed, and the associated (weighting) values of the transition costs are specified.

Given the difficulties encountered in locating automatically, accurately, and in real-time, the endpoints of a test sequence, it is advisable to relax the endpoint constraints by permitting the path to start from one of the following nodes: $(1, 1)$ to $(1+\epsilon_{i_1}, 1)$, or $(1, 1)$ to $(1, 1+\epsilon_{j_1})$, and to end at one of the following nodes: $(I-\epsilon_{i_2}, J)$ to (I, J), or $(I, J-\epsilon_{j_2})$ to (I, J).

Figure 8 shows some examples of local (left) and accumulated (right) distance matrices for similar (top) and different (bottom) gestures. Low distance values (depicted by blue pixels) are obtained when comparing similar gestures. Conversely, high dissimilarities are observed when the tested gesture is very different from the reference one, resulting in a worse matching score.

4.1.2. Real-time Max/MSP implementation of the Multigrid DTW algorithm

In the context of real-time use, without pre-segmentation, we implemented a method using simultaneously a set of shifted DTW grids, each one hypothesizing another starting region for the tested input, where the hop size (or time shift) between two successive DTW grids will generally be higher than or equal to $(1 + \epsilon_{i1})$. The number of simultaneously active grids can be limited to the following quantity: $S_{max} = ceil(I_{max}/hop\_size)$ where I_{max} can be determined according to the global and/or local constraints, and

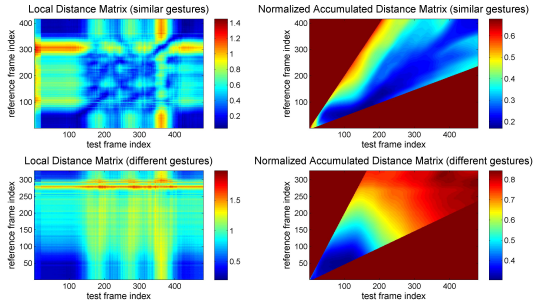

Figure 8. Local (left) and Accumulated (right) Distance Matrices for similar (top) and different (bottom) gestures

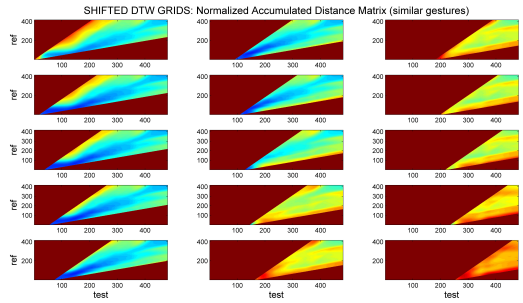

Figure 9. (Normalized) Accumulated Distance Matrices for successive shifted DTW grids (similar gestures)

depends on the reference gesture length. After appropriate initializations (e.g. setting all accumulated distances to "infinite" values), each shifted grid will output its (current) best score at any time instant. The minimum value of all these normalized accumulated distances will be assigned to the given reference gesture. Moreover, one can get an approximate value of the expansion or compression ratio, knowing the terminal node exactly and the first node coordinates approximately (depending on ϵ or hop_size parameters), without the need of keeping trace of the origin throughout the whole grid, as well as without backtracking.

Figure 9 illustrates (normalized) accumulated distance matrices for successive shifted DTW grids when test and reference gestures are similar. A good matching score is obtained for the low shift values, while it becomes worse when the delay increases.

4.2. Data preparation and Distance Metrics

In the current version of our system implements a downsampling stage (with a ratio of 4), preceded by a lowpass filtering step. However, we plan to investigate some more preprocessing: for example, removing the gravity component, developing tilt-invariant features, combining differently left/right accelerometer/gyroscope data, etc.

When computing the local distance, each sensor axis can be weighted independently so as to (a) compensate for different ranges of values (e.g. accelerometer data $\pm 2g$ and angular velocity $\pm 500°/s$) and (b) compare recognition scores when emphasizing different types of sensors. As each test or reference frame is characterized by a N-tuple of values, it can be viewed as one point in an N-dimension space and

classical distance measures can be performed: e.g. the L^1 distance, whose computation is very efficient as its expression is made of a (weighted) sum of the absolute value of differences (neither square, nor square root computed).

As we use the angular velocities and the accelerations (and not positions), the "amplitudes" of these sensor data are also affected by the speed of the gesture execution. One might prefer other computations of the local distance or dissimilarity measure (Mahalanobis distance, cosine correlation coefficient, ...), but at the price of higher computational load. Also, accelerometer and gyroscope data could be treated separately, and even differently. Note for example that the coordination between the movements of both legs must be preserved in our current approach: if the movement of one leg is delayed with respect to the other one, the gesture might become unrecognizable.

4.3. Post-processing

The only currently implemented post-processing consists in selecting, at each moment, the gesture with the lowest normalized accumulated distance and validating its recognition if this value is below a global threshold. However, the developed recognition tool is adapted to the evaluation of a confusion matrix, which can give information about eventual ambiguity between different pre-defined gestures. The decision criteria might be improved by using gesture-based threshold values, as well as by preventing the erroneous detection of "overlapping" gestures. Finally, a possible interaction between the recognizer and the performance could be implemented by means of an N-best strategy, using the N-best scores to interpolate the parameters of the sound effects.

5. Mapping by Interpolation

Since interpolation between sets of parameters was proposed by Allouis [5] for the SYTER at GRM, several implementations [6, 7, 8] have been developed. Interpolation is an operation whereby each value of the output set of m values is a weighted sum of the corresponding values in the n interpolated sets, with normalized weights so that $\sum W_{Ni} = 1$:

$$output\_val_j = \sum_{i=1}^{n} W_{Ni} \ val_{ij} \quad 1 \leq j \leq m \qquad (2)$$

As such, it can perform the three types of mapping usually described in the literature (direct or one to one, divergent or one to many, convergent or many to one) depending on the number m of values in each set and on the dimensionality of the interpolation space, which is not to be confused with the number n of points placed within that space. We don't mean that it should or could replace other mapping techniques, but its intuitiveness makes it a very valuable tool.

5.1. Gravitational field metaphor

Allouis [5] used that metaphor where each of the n points exerts an attraction force on the cursor depending on their distance d_i. Associating a weight proportional to this force to each point i and representing its mass by its radius R_i, we obtain the equation ($power = 2$ for Newton's law):

$$W_i = \frac{R_i}{max(d_i, dmin)^{power}} \quad 1 \leq i \leq n \qquad (3)$$

where $dmin$ is a small number that prevents a division by zero when $d_i = 0$ and is defined inside the external as $10^{(\frac{-6}{power})}$ to guarantee that the maximum weights do not depend on the $power$ factor.

The user may use the Max message `power $1`, with a float argument, hereby modifying the shape of the interpolation between points: softer in the vicinity of the center with low values of $power$, more abrupt with higher values. Other radial basis functions (RBF) could also be used.

5.2. Rmin

Interpolation tools are traditionally controlled with a mouse. As we planned to use interpolation with sensors data input, where a precise location is quite difficult to attain by the performer, dancing on stage without visual feedback from the computer screen, we decided to define a circular zone of radius $Rmin$ around the interpolation points where the maximum weight is always reached. A negative $Rmin$ would on the other hand prevents the user from reaching the mathematical centre closer than the absolute value of $Rmin$. This allows to define points that modify the values in their vicinity while never taking over, as they can never reach the maximum weight. Their influence decrease as their $Rmin$ reaches more negative values.

For the sake of clarity, we decided to bound the absolute value of $Rmin$ by R, so that the outer circle always displays R. To avoid the same type of circles for R and $Rmin$, which could be confusing when points are superposed, we choose to display Rmin with a filled circular area. The filled area includes the centre for a positive $Rmin$, showing the zone where the maximum weight is reached. It surrounds the centre at $Rmin$ distance for a negative one, showing the additional distance to the mathematical center .

5.3. The Void

The idea of perturbation made possible by negative $Rmin$ led to the concept of a constant field over the whole space, the void, linked to its own set of values. Within this constant field, obtained simply by attributing a user-defined constant weight to the void, the points would perturb the field with their associated sets of values. But we also defined a weight depending on the combined distance of the cursor to all the other points. We then compute the weight of the void as the sum of those two distinct contributions.

5.4. Max external

Instead of defining a new GUI object, we decided to make a Max external that generates all necessary graphical commands to existing GUI objects: *LCD* or *jit.pwindow* (using

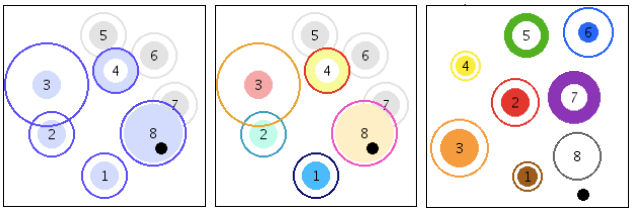

Figure 10. The 3 color modes, inactive points are light grey

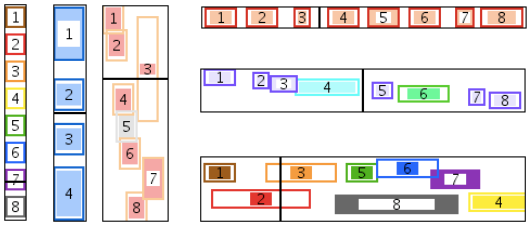

Figure 11. Several 1-D setups in different color modes

jit.gl.sketch and *jit.gl.render*). Sprites are defined for each point and for the current point at instantiation of the external object. They are redefined each time a point is resized or (de)activated. The communication is bidirectional as the GUI also returns mouse position, mouse state (clicked or idle), and modifiers in order to move, (de)activate, and resize the points. Creation arguments of the module are the number n of interpolation points, the number m of values for each interpolation set associated to a point as well as both the horizontal and vertical size of the GUI object in pixels. Colors can be defined with the RGB output of the *swatch* object. The different available color modes are shown in Figure 10 for a 2-D interpolator and, in Figure 11, for various vertical or horizontal 1-D versions, using *LCD*:

- same_colors: all points i share the same R_i color and the same $Rmin_i$ color.
- individual_colors: each R_i and $Rmin_i$ colors can be defined individually.
- resistor_colors: users familiar with the resistor code can locate the points faster, without having to read the numbers.

6. Conclusion

We should further investigate the possibilities offered by the wavelet analysis, including an more efficient way to define the best matrix of weights using a database of hits. The DTW showed good results with two databases: one of dance gestures and one of hand gestures. But we are investigating further improvements of the recognition scores by adding non-linear quantification, finding the optimal weights, and refining the post-processing. The presence of the time ratio between the reference and performed gestures in the results may be used by the post-processing algorithm but can also serve as feedback to the performer and as an incitation to record some gestures at different speeds if the reference speed is rarely matched. The usual 2-D interpolation object was fully integrated in the ARTeM framework and $Rmin$ made it possible to reach the points navigating the space with sensors without visual feedback. The 1-D version proved also very useful to attach, for instance, various transposition ratios on the fly at different positions of a limb and define their transitions. We are currently finishing the *jt.pwindow* implementation of a 3-D interpolation objet.

7. Acknowledgments

Research supported by `numediart`, a long-term research program centered on Digital Media Arts, funded by Région Wallonne, Belgium (grant N°716631). With the help of the Ensemble Musiques Nouvelles [4], Mons, Belgium.

References

[1] http://www.numediart.org/blog/?cat=9
[2] http://www.quartetproject.net/space/Program
[3] http://www.michele-noiret.be/index.php?page=de-deux-points-de-vue
[4] http://www.musiquesnouvelles.com
[5] J. Allouis and J. Y. Bernier, "The SYTER project: Sound processor design and software overview". *In Proc. ICMC*, pp. 232-240. 1982.
[6] T. Todoroff, C. Traube, and J. M. Ledent, "NeXTStep graphical interfaces to control sound processing and spatialization instruments". *In Proc. ICMC*, pp. 325-328, 1997.
[7] Momeni, Ali and Wessel, David "Characterizing and Controlling Musical Material Intuitively with Geometric Models". *NIME*, McGill University, Montreal Canada, 2003.
[8] R. Bencina, "The metasurface: applying natural neighbour interpolation to two-to-many mapping". *In Proc. NIME*, pp. 101-104, 2005.
[9] T. Todoroff, E. Daubresse, and J. Fineberg, "Iana : a real-time environment for analysis and extraction of frequency components of complex orchestral sounds and its application within a musical realization". *In Proc. ICMC*, pp. 292-293, 1995.
[10] J. Deller, J. Proakis, and J. Hansen, "Discrete-Time Processing of Speech Signals", Macmillan, 1993.
[11] S. Mitra and T. Acharya, "Gesture Recognition: A Survey", IEEE Trans. on Syst., Man, and Cybernetics Part C, Vol. 37, No. 3, pp. 311-324, May 2007.
[12] C. Myers, L. R. Rabiner, and A. E. Rosenberg, "Performance Tradeoffs in Dynamic Time Warping Algorithms for Isolated Word Recognition", *In IEEE Trans. Acoust., Speech, Signal Processing*, vol. ASSP-28, No.6, Dec. 1980.
[13] H. Sakoe and S. Chiba, "Dynamic Programming Algorithm Optimization for Spoken Word Recognition", *In IEEE Trans. Acoust., Speech, Signal Processing*, vol. ASSP-26, pp. 43-49, Feb. 1978.
[14] S. G. Mallat, "A theory for multiresolution signal decomposition: the wavelet representation", *EEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 11, pp. 674-693, 1898.
[15] M. V. Wickerhauser, "NRIA lectures on wavelet packet algorithms", *Lecture notes, INRIA*, pp. 31-99, 1991.