# MnM: a Max/MSP mapping toolbox

Frédéric Bevilacqua, Rémy Müller and Norbert Schnell

Real Time Applications Team
Performance Arts Technology Group
Ircam Centre Pompidou
1 pl. Igor Stravinsky
75004 Paris – France
+ 33/1 44 78 48 43
{Frederic.Bevilacqua, Remy.Muller, Norbert.Schnell}@ircam.fr

## ABSTRACT

In this report, we describe our development on the Max/MSP toolbox MnM dedicated to mapping between gesture and sound, and more generally to statistical and machine learning methods. This library is built on top of the FTM library, which enables the efficient use of matrices and other data structures in Max/MSP. Mapping examples are described based on various matrix manipulations such as Single Value Decomposition. The FTM and MnM libraries are freely available.

## Keywords

Mapping, interface design, matrix, Max/MSP.

## 1. INTRODUCTION

In gesture controlled digital audio systems, the term "mapping" refers to the relationship between the gesture/control data and the digital sound processes. The ability to choose and control this relationship makes digital instruments fundamentally different than acoustic or electric instruments.

Different types of mapping have been developed over the years, often using idiosyncratic methods. Recently, mapping methods have been the subject of formalization and discussed in several papers [1]-[9]. Generally, mapping strategies are separated in three different classes: *one-to-one*, *one-to-many*, *many-to-one*. By combining these classes, *many-to-many* mappings can be built. Interestingly, it has been recognized that such complex mappings are more satisfactory, after a learning phase, than one-to-one mappings [2].

Nevertheless, there is still a lack of practical tools to implement complex mappings in a relatively intuitive manner. For this reason, we are currently developing a series of Max/MSP externals and abstractions, the MnM toolbox, based on the free FTM library.

The goal of this paper is to present our approach for this ensemble of Max/MSP externals and abstractions, based on modular matrix manipulations. We describe here the use of a first set of objects available with help patches[2].


*Nime'05*, May 26-28, , 2005, Vancouver, BC, Canada.


## 2. RELATED WORKS

Mapping strategies have been reviewed in recent papers and we refer the reader to references [1] and [2] for a comprehensive overview.

Van Nort et al. [9] give a mathematical formulation of mapping as a function *g* between a controller parameter space $\Re^n$ and a sound parameter space $\Re^m$. If the mapping is described by a series of discrete couples of vectors $\{X_i, Y_i\}$, where $X_i \subset \Re^n$ (control parameter space) and $Y_i \subset \Re^m$ (sound parameter space), the mapping can be seen as an *interpolation* problem. Van Nort et al. [9] and Goudeseune [4][5] have provided elegant mathematical solutions for such an approach.

It is interesting to note that if the series $\{X_i, Y_i\}$ overdetermines the mapping function, considering a possible uncertainty for each value of the vectors $\{X_i, Y_i\}$, the problem can be then considered as a *regression* problem.

Finally, mapping procedures can be also viewed as pattern *recognition* problems, especially *many-to-few* mappings. For example, neural networks have been previously used, as described in [10][11][12].

Nevertheless, other common linear and non-linear techniques in statistical and machine learning methods seem promising for mapping, for example Principal Component Analysis, Linear Discriminant Analysis, Gaussian Mixture Models, Kernel Methods and Support Vector Machines, Hidden Markov Models. Even if particular cases have been implemented in dedicated Max/MSP externals [17][18], the use of such methods remains generally cumbersome in real-time musical context. The long-term goal of the MnM project is to fill this gap by providing general and modular mapping tools.

## 3. MAPPING USING MATRICES

### 3.1 Basic operations

A first approach consists in building mapping procedures as a combination of relatively simple matrix operations. Consider X, a vector of size n from the controller parameter space and Y, a vector of size m from the sound parameter space. A simple

[2] http://recherche.ircam.fr/equipes/temps-reel/maxmsp/mnm.html.

mapping operation corresponds for example to the matrix multiplication with a m×n matrix A.

$$Y = A*X \quad \text{Eq. 1}$$

A is a *n-to-m* linear mapping (from $\Re^n$ to $\Re^m$). This formulation can include both cases n≥m (*many-to-few*) or n<m (*few-to-many*). Note that A can be defined as time-dependant, i.e. A(t).

If n≥m, A is a projection from the Euclidian space $\Re^n$ to an hyperplan in a subspace $\Re^m$. If m≥n, A can be interpreted as a linear extrapolation by defining a hyperplan in a space of higher dimension.

The matrix A can be exactly determined by a series of n examples $\{X_i, Y_i\}$, where $1 \le i \le n$. If the number of example is larger than n, A can be determined by linear regression. The solution for this case will be described in the section 5.

More complex mappings can be built as a combination of several matrices. In particular, a layered mapping [7] can be easily set by a series of matrix multiplications. For example, considering three matrices A, B and C, the mapping can be easily defined as:

$$Y=(A*B*C) *X \quad \text{Eq. 2}$$

In particular, note that the intermediate mapping layer B can operate in a space of different dimension than n or m.

Other matrix operations can be also considered, such as the element by element multiplication (noted here as **.***). In such a case, the dimensions of the matrices A and B must be identical.

$$Y=(A.*B) *X \quad \text{Eq. 3}$$

A matrix can also be combined with a function f applied to each element of the matrix.

$$Y= [f(a_{ij})]*X \quad \text{Eq. 4}$$

The obvious interest of such an operation is to introduce non-linear mappings. As a matter of fact, the combination of equations 2, 3, and 4 enables the design of powerful non-linear mappings (which can have similar structures to neural networks).

## 3.2 Notation

The case of an affine transform (B is a vector of size m×1):

$$Y = A*X + B \quad \text{Eq. 5}$$

can rewritten as an extension of A:

$$Y = A_e *X_e \quad \text{Eq. 6}$$

where $A_e$ is a m×(n+1) matrix.

The vector $X_e$ of size n+1 is built from the vector $X= (x_1,\ldots, x_n)$ as follows:

$$X_e = (x_1,\ldots, x_n, 1) \quad \text{Eq. 7}$$

From this point, we will use this notation that allows for a more compact representation of the mapping. In this case, the number of examples necessary to completely determined $A_e$ is n+1.

## 3.3 Single Value Decomposition (SVD)

Obviously several other matrix operations can be useful to extend the mapping procedure. In particular, Singular Value Decomposition allows for the computation of the inverse (or pseudo-inverse) of a matrix, linear regression and Principal Component Analysis [14].

We briefly recall here SVD. Possible applications are commented further in section 5.

Consider a n×m matrix M. The SVD decomposition corresponds to compute three matrices U, S, V whose sizes are n×m, m×m, and m×m, respectively:

$$M = U * S * V^t \quad \text{Eq. 8}$$

U and V are unitary matrices (i.e. $U^t=U^{-1}$). S is diagonal, and its elements are ordered in decreasing values.

# 4. Max/MSP IMPLEMENTATION

We chose to develop mapping modules using the recent shared library FTM in Max/MSP, which enables matrix handling. We describe FTM shortly in the next section. The integration of the approach shown in this report could be easily performed in other software with equivalent matrix structures and methods.

## 4.1 FTM

FTM is a shared library for Max/MSP providing a small and simple real-time object system and optimized services to be used within Max/MSP externals. FTM is distributed under LGPL[3].

The main purpose of FTM is the representation and processing of sound, music and gesture data in Max/MSP extending the data types processed and exchanged by the Max/MSP modules. The implemented classes include matrices, dictionaries, sequences, break point functions and tuples.

FTM allows for static and dynamic creation of complex data structures. An extended Max/MSP message box allows for the evaluation of arithmetic expressions, function calls and method invocation on FTM objects.

FTM objects can contain references to other FTM objects. A simple garbage collector handles transparently the destruction of dynamically created FTM objects referenced by multiple elements of a patch. FTM supports MIDI and SDIF file formats.

The FTM *fmat* class implements a simple two-dimensional matrix of floating-point values providing methods for inplace matrix calculations and data import/export. An FTM *track* object allows for recording and playing of a stream of matrices as well as for the import/export of a stream in the SDIF file format.

The *mat* class acts as a 2-dimensionnal cell array of generic FTM objects, and in particular can handle matrices of *fmat.*

The externals of the libraries based on FTM use *fmat* as a generic representation for a variety of algorithms implementing analysis/synthesis, mapping, statistical modeling, machine learning and information retrieval. FTM enable to easily connect these algorithms in an application, thus creating a tied link between gesture analysis and sound synthesis.

## 4.2 MnM

MnM, "*Mapping is not Music",* is a set of Max/MSP externals based on FTM, taking advantages principally of the matrix classes *fmat* and *mat*. The construction of the mapping procedure is performed using both basic matrix operations from the FTM library and using the dedicated MnM set of externals and abstractions. As already stressed, mapping can be thus built in a modular way. Different types of mapping

[3] http://www.ircam.fr/ftm

approaches, including interpolation, regression and recognition are implemented.

## 5. EXAMPLES

We explain here two abstractions, mnm.matmap and mnm.pca, which illustrate the use of the external mnm.svd (performing the Single Value Decomposition of a matrix).

### 5.1 mnm.matmap

The idea of this abstraction is to implement a multidimensional linear mapping. It can be seen as basic module to build complex *n-to-m* mapping.

The mapping is based on the matrix multiplication $Y=A_e*X_e$, as described in Eq.6, corresponding to an affine transform.

The matrix $A_e$ can be determined by a set of $k$ mapping examples $\{X_i, Y_i\}$, called "training examples". Two matrices are created from these examples:

1) $X_{train}$, of size $(n+1)\times k$, formed by concatenating the vectors $X_i$

2) $Y_{train}$, of size $m\times,k$ formed by concatenating the vectors $Y_i$. The concatenation is performed by the object mnm.q (Fig.1)

We can therefore write the following equation:

$$Y_{train} = A_e * X_{train} \qquad \text{Eq. 9}$$

A SVD decomposition of the matrix $X_{train}$ enables the determination of A as shown below, (taking advantage of the fact that U and V are unitary matrices):

$$\text{SVD: } X_{train} = U*S*V^t \qquad \text{Eq. 10}$$

$$\Rightarrow Y_{train} = A_e*U*S*V^t \qquad \text{Eq. 11}$$

$$\Rightarrow Y_{train} * V*S^{-1}*U^t = A_e \qquad \text{Eq. 12}$$

The computation of $S^{-1}$ is simple since S is diagonal. Note that $V*S^{-1}*U^t$ corresponds to compute the inverse of $X_{train}$,, if this latter exists. For the other cases, in particular if $k\neq(n+1)$, the SVD procedure still guarantees the determination of A (corresponding to a pseudo-inverse).

This procedure can be easily performed in Max/MSP thanks to the MnM objects mnm.svd and mnm.xmul, as shown in Fig.1. For example, the abstraction mnm.matmap allows for the computation of the matrix (bottom part of Fig1), as well as the multiplication $Y=A_e*X$. The matrix $A_e$ can be imported/exported as a txt file.

The top part of Fig1 shows a possible use of mnm.matamp, i.e. the interpolation between different waveforms controlled by the 2D positions of the cursor.

Several of these objects can be used in parallel, enabling piecewise linear mappings [9][13]. Note also that switching and/or interpolating between matrices in real-time, allows for interesting "dynamic" mapping procedures.

### 5.2 mnm.pca

Similarly to the previous section, consider the matrix $X_{train}$ (size $n\times k$) formed by concatenating a series of $k$ vectors of the controller space $\Re^n$. The mnm.pca object performs Principal Component Analysis (PCA), based on the SVD computation, that outputs the three matrices U, S, and V. The first p principal components of $X_{train}$ are determined by keeping only the first p diagonal elements of S (setting the others to zero). Note that $X_{train}$ must be centered prior to performing SVD.

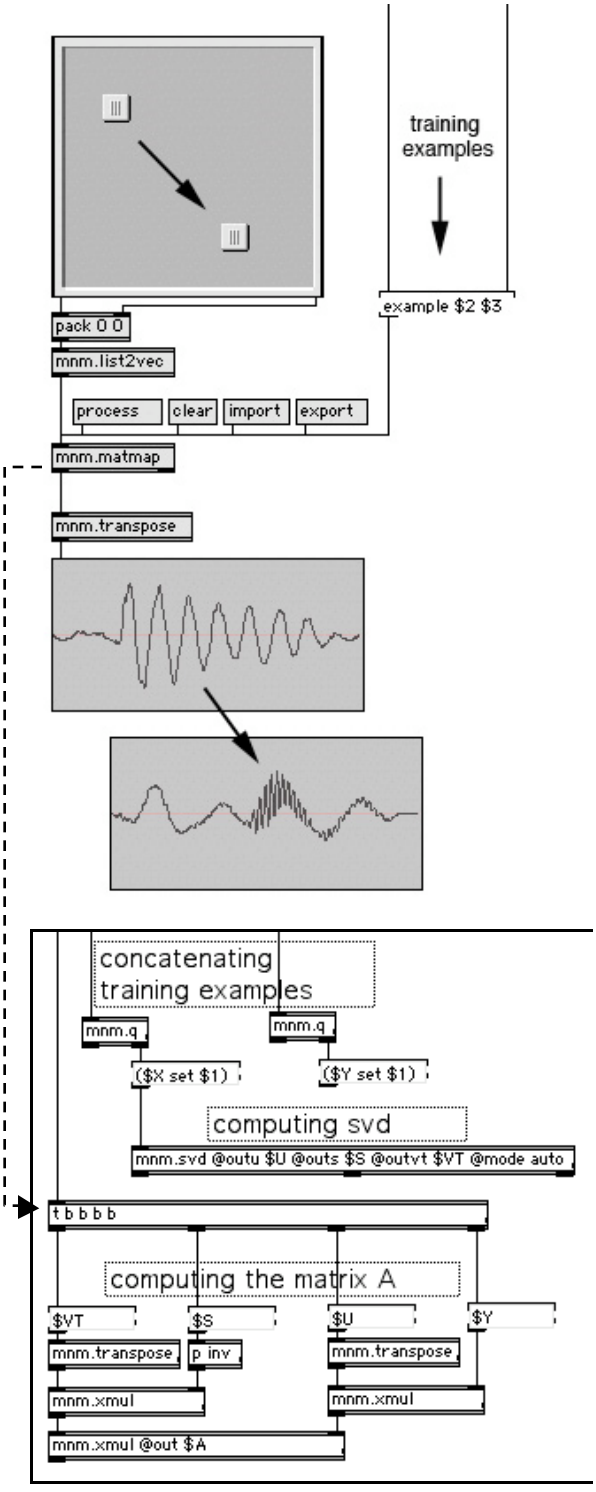


**Fig.1 Max patch using the abstraction mnm.matmap, illustrating a 2 to 600 mapping, based on a set of training examples. The bottom part of the figure shows the part of mnm.matmap where the mapping matrix is computed.**

Thus, PCA enables the reduction of the dimension of the effective control space, which can simplify the mapping procedure [15]. As a matter of fact, the actual space dimension formed by all controller values is often lower than $n$. This occurs typically when some configurations cannot be played due to physical constraints. In such a case, PCA can be used to define a new orthogonal basis of the actual controller space.

PCA can also be seen as a practical way to parameterize principal features of the control space (or the sound parameter space). After decomposing $X_{train}$ in principal components, the re-synthesis is possible using the matrices U, S, and an additional "control" vector C (size of p×1):

$$X = U*S*C \qquad \text{Eq. 13}$$

For example, the first value of C weights the major component of this space, whereas the last one weights the smallest component, generally a "noise" contribution.

## 5.3 Towards kernel methods

Principal component analysis based on SVD can suffer from severe limitations due to the assumption of the linear combination of the components. Nevertheless, kernel methods allows for the extension of PCA to non-linear problems, called Kernel-PCA [16]. We are currently investigating such an approach for mapping, which will be implemented in MnM in the near future.

# 6. DISCUSSION AND PERSPECTIVES

We described our approach for the design of mapping tools and some elements of the MnM library. The modular design of the MnM library greatly facilitates the experimentation of various mapping strategies, including interpolation, regression and recognition.

The object we described, mnm.matmap and mnm.pca were found to be very simple to use and promising. Their main limitation resides in the fact they model data linearly. However, as already mentioned, such objects can be generalized for non-linear mapping using kernel methods.

# 7. ACKNOWLEDGMENTS

We would like to thank Riccardo Borghesi, Diemo Schwarz, Emmanuel Fléty, Nicolas Leroy and Nicolas Rasamimanana for providing a fertile environment for this project and Suzanne Winsberg for interesting discussions on statistical methods. Special thanks to Matthew Burtner for his enthusiasm and for being the first user of matmap in a concert.

# 8. REFERENCES

[1] Wanderley, M and Battier, M -editors. *Trends in Gestural Control of Music*. IRCAM, Centre Pompidou, 2000.

[2] Wanderley, M. -editor, Mapping Strategies in Real-Time Computer Music, Organised Sound, 7(2), 2002.

[3] Choi, I., R. Bargar, and C. Goudeseune. A manifold interface for a high dimensional control space. *Proc. of the Int. Computer Music Conf.*: 385–92, 1995.

[4] Goudeseune, C. 2001. *Composing with parameters for synthetic instruments.* DMA thesis, Urbana-Champaign, IL: University of Illinois.

[5] Goudeseune, C. Interpolated Mappings for Musical Instruments, *Organised Sound* 7(2), 2002.

[6] Hunt, A., and R. Kirk. Mapping strategies for musical performance. In Wanderley and Battier (eds.) *Trends in Gestural Control of Music*. Paris: IRCAM, Centre Pompidou, 2000.

[7] Hunt, A., M. Wanderley, and R. Kirk. Towards a model for instrumental mapping in expert musical interaction. *Proc. of the Int. Computer Music Conf.*: 209–12, 2000.

[8] Rovan, J., M. Wanderley, S. Dubnov, and P. Depalle. Instrumental gestural mapping strategies as expressivity determinants in computer music performance. *Kansei, the Technology of Emotion. Proc. of the Associazione di Informatica Musicale Italiana Int. Workshop*: 68–73, 1997.

[9] Van Nort, D., Wanderley, M. M., Depalle, P. On the Choice of Mappings Based On Geometric Properties, *Proc. of the International Conference on New Interfaces for Musical Expression,* 2004.

[10] Cont, A., Coduys, T., Henry, C.. Real-time Gesture Mapping in Pd Environment using Neural Networks, *Proc. of the International Conference on New Interfaces for Musical Expression,* 2004.

[11] Fels, S. S. and Hinton, G. E. Glove-Talk: A neural network interface between a data-glove and a speech synthesizer. IEEE Trans. On Neural Networks, vol. 4, No. 1, 1993.

[12] Lee, M., Freed, A., Wessel, D. Real-Time Neural Network Processing of Gestural and Acoustic Signal, *Proceedings of the 17th International Computer Music Conference*, Montreal, 1991.

[13] Bowler, I., P. Manning, A. Purvis, and N. Bailey. On mapping *n* articulation onto *m* synthesiser-control parameters. *Proc. of the Int. Computer Music Conf.*: 181–4, 1990.

[14] Turk, M. A random walk through eigenspace, *IEICE Trans Information and Systems*, 2001.

[15] Bevilacqua, F., Ridenour, J., Cuccia, D. J. 3D motion capture data: motion analysis and mapping to music, *Proceedings of the Workshop/Symposium on Sensing and Input for Media-centric Systems*, University of California Santa Barbara, 2002.

[16] Schölkopf, B., Smola, A, and Müller, K.-R.. Kernel principal component analysis. In B. Schölkopf, C. J. C. Burges, and A. J. Smola, editors, Advances in Kernel Methods - SV Learning, pages 327-352. MIT Press, Cambridge, MA, 1999.

[17] Kolesnik, P. and Wanderley, M. Recognition, Analysis and Performance with Expressive Conducting Gestures. In *Proceedings of the 2004 International Computer Music Conference (ICMC2004)*, Miami, Fl., 2004.

[18] Momeni, A. and D. Wessel, Characterizing and Controlling Musical Material Intuitively with Graphical Models, *Proceedings of the New Interfaces for Musical Expression Conference*, Montreal, Canada, 2003.