jpb.mod: a Max/MSP data modification library for rapid prototyping

Jon Bellona
Virginia Center for Computer Music
(VCCM)
University of Virginia
Charlottesville, VA
jon@jpbellona.com

ABSTRACT

This paper presents a Max/MSP library that addresses data modification (e.g. sensor conditioning) through a rapid-prototyping approach. Drawing upon digital instrument design context, we reframe common data modifications frequented in the musical and digital art compositional process. We consider existing mapping systems and literature in order to catalog data modification primitives with goals of creating modular data modification tools (*jpb.mod*). The *jpb.mod* library, a Max 6 package built around these primitives, handles the modification of a singular data stream for rapid prototyping. The *jpb.mod* data modification library draws upon the theory and principles of UX and UI design, considering both efficiency and usability.

Author Keywords

Max/MSP, data modification, Max 6 package, UX, rapid prototyping

ACM Classification

H.5.2 [Information Interfaces and Presentation] User Interfaces-Graphical user interfaces (GUI), H.5.2 [Information Interfaces and Presentation] User Interfaces-Prototyping, D.2.2 [Software Engineering] Design Tools and Techniques-Modules and interfaces, H.5.5 [Information Interfaces and Presentation] Sound and Music Computing.

1. INTRODUCTION

"Everything should be made as simple as possible, but not simpler." - Albert Einstein

Existing studies conclude that complex mapping systems afford more complex possibilities [1][2][3]. If complex mapping systems are to be developed, used, and refined, our compositional process should support the generation of complex mapping relationships with relative ease and flexibility. Existing toolkits, libraries, and mapping systems do offer solutions to various mapping problems (e.g. interpolation [1][4], input/output [5][6], and data modification [5][7]). Yet, these solutions do not resoundingly solve the challenges of developing complex mapping systems. We explore rapid prototyping as an alternative strategy. By applying a rapid prototyping framework on one particular subset of mapping—data modification—we define and design tools for aiding the creation of complex mapping relationships.

NIME'15, May 31-June 3, 2015, Louisiana State Univ., Baton Rouge, LA.

1.1 Rapid Prototyping

In industry, the role of rapid prototyping is to "compress the development cycle in order to take advantage of growth" [8]. For digital music application, we borrow the rapid prototyping process common to iteration steps in design thinking [9]. Applying rapid prototyping to the creation of music software tools may result in a deepening of the musical creation process—more time spent listening, developing aesthetics and concepts, acquiring user feedback, and refining core system elements. By addressing needs for increased time in responding to sonic results, rapid prototyping may help refine how we approach complex-mapping problems.

1.2 Mapping

We chose to develop core, rapid prototyping tools essential to mapping. Mapping has been "commonly defined as the translation layer correlating gesture to sound" [10]. Starting under the guise of digital musical instrument design, mapping is but one of an instrument's three elements [3]:

* gesture input
* mapping
* sound production output

Breaking down mapping into an element list reveals yet another triadic set, one that includes data modification.

* data input
* data modification
* data output

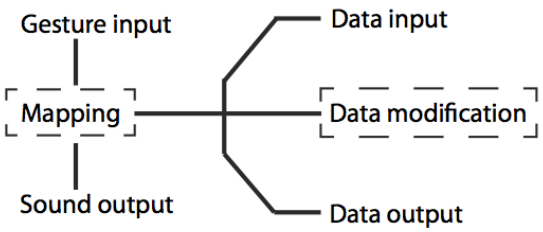

Figure 1: Elements of a digital musical instrument definition highlighting data modification as part of the mapping process.

2. DATA MODIFICATION

Because data modification is a process used frequently within mapping (e.g. sensor conditioning), we chose to focus on developing data modification tools. The term "data modification" is what the name implies: a transformation of data, which may be used in preparation for or in conjunction with control assignment. In fact, dedicated topics on mapping strategies contend that most input data requires some modification before the data may be used to control parameters and/or sound synthesis [11][12][13].

Figure 2: Signal flow diagram of digital data modification as part of the mapping process.

2.1 Data Modification Example

A basic example of data modification may further help outline its essential nature. (Table 1) details the mapping of a 0-1 floating-point number onto a chromatic scale.

Table 1: Data Modification Example

Data input	0-1 floating-point numbers.
Data modification	Scale 0-1 to 0-12.
	Thin floating-point numbers to whole numbers.
	Offset 0-12 by 60 to 60-72.
Data output	Output 60-72 as MIDI nn, a parameter of pitch (C4-C5).

2.2 Data Modification Types

Sensor-based systems may incorporate data modification as part of their software [14][15][16][17]. For example, Tanaka and Donnarumma apply normalization, interpolation, and filtering of MMG/EMG sensor data before mapping [18]. Other systems modularize data modification as a way to enable users [5][7]. Often, data modification modules remain rigidly contained within their own systems [15][16]. Regardless, similarities abound. With special acknowledgment to Dr. Jeffrey Stolet, whom the author first heard about the following types, the author proposes Stolet's five data modification types, which articulate the process of modifying a one-dimensional digital data stream. The five types are:

1. Interpolation
2. Thin
3. Offset
4. Smooth
5. Scale

The five data modification types may function at control or signal rate. Modification types may include nonlinear transform functions (e.g. interpolate, scale) and may be used to generate event-based triggers (e.g. interpolate, thin). The five data modification types are combinatorial, and may be placed in varying orders to achieve both simple and complex results, as will be shown below. Because other data modification types may arise, these five types should be viewed as a starting point for mapping research and tool development.

2.2.1 Interpolation

Interpolation, broadly speaking, produces an estimated numerical location based upon the dependence of surrounding locations [19]. One may view interpolation as the generation of new values (estimated numerical location) that sit between discrete points (surrounding locations) found within a one-dimensional data stream.

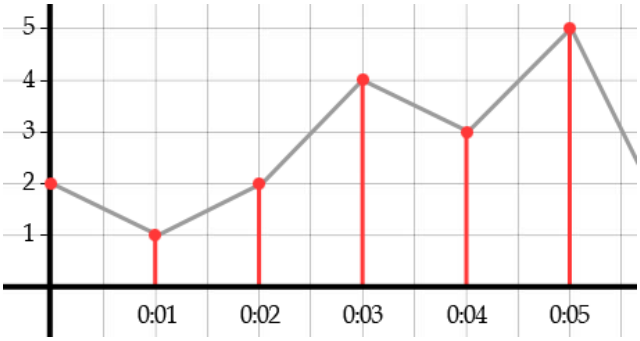

Figure 3: The line between points represents a linear interpolation between discrete points over one second time intervals.

2.2.2 Thin

Thin, similar to certain filters, "removes some component or characteristic of a signal" [20]. Thinning data often uses one of three functions: thin as a function of time (e.g. poll/sample) [4], thin as a function of equalization (e.g. linear filters) [18], or thin as a function of thresholds (e.g. hysteresis) [21].

2.2.3 Offset

Offset modifies data through basic addition or subtraction. While offset appears to be a simple modification, composers frequently transpose (offset) pitch to develop material; one common example is the musical sequence [22].

2.2.4 Smooth

Smooth algorithmically modifies data to maintain patterns and to help reduce noise [23]. While smoothing algorithms are operations found in FIR filters, which one may conceive of as thinning, a different conceptual model and algorithms necessitate a unique modification type [24].

2.2.5 Scale

Scale is a basic function of mathematical ratios: multiplication and division. Variants of this function may be found in existing sensor-based systems [16][17].

2.3 Data Modification Examples

To help outline the five data modification types, we may look to existing systems. The examples below (Xth Sense, simpleKinect, eMersion) display how data modification types may be combinatorial.

Table 2: Xth Sense (Biophysical Instrument) [15]

Data input	Audio signal
Data modification	Low-pass EQ (thin)
	Normalize between -1.0 and 1.0 (scale)
Data output	Output to data extraction module

Table 3: simpleKinect (Kinect-to-OSC application) [25]

Data input	Joint vectors from a video matrix signal
Data modification	Reduce joint vector jitter over three frames at 60 f.p.s. (smooth)
Data output	Output joint vector as OSC message

Table 4: eMersion (Wireless Performance Technology) [16]

Data input	Serial data signal
Data modification	Reduce analog sensor noise through averaging (smooth)
	Convert data stream to 0-1 for user mapping (scale)
Data output	Output data (specified by user)

Given the widespread use of data modification, developing rapid prototyping tools for data modification helps serve mapping projects and complex mapping problems.

3. JPB.MOD

jpb.mod is a Max 6 package created for rapid prototyping. Each module addresses one of the five data modification types (interpolate, thin, offset, scale, smooth [itoss]). *jpb.mod* objects handle the modification of a one dimensional data stream functioning at control rate.

3.1 Max/MSP

In software, we may consider UI as comparable to rapid prototyping a 3D product, one that puts "physical handles on phantom models" [11]. Based upon the user interface (UI) design tools and the popular support within computer music and digital art fields, we chose to develop the *jpb.mod* tools in Max/MSP, focusing upon modular design, accessibility, affordance, and constraints [26].

3.1.1 Max 6 Rapid Prototyping Example

An input module that comes standard with Max 6, *demosound.maxpat*, serves as an example of a rapid prototyping tool (Figure 4). The module encapsulates various audio generators within a single *bpatcher* object. While hiding some functions under the hood, the GUI enables easy access to device controls, and the module handles most audio generation tasks within Max/MSP.

3.2 jpb.mod Rapid Prototyping Tools

jpb.mod modules take advantage of Max system strengths, namely UI, scripting, and presets (*pattr*) in order to support different user behaviors, such as real time and scripting use. The *jpb.mod* tools may be created either as *bpatcher* objects or as Max abstractions (UI functions may still be accessed by double clicking the abstraction). Each *jpb.mod* object has a help file and help reference menu, and nearly all *jpb.mod* functions offer a monitor window to visually report data streams. The description of each tool below will further outline additional strengths.

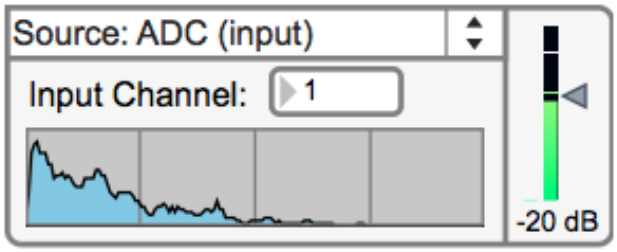

Figure 4: *demosound.maxpat* is an example of a rapid prototyping tool developed by Cycling74.

3.2.1 Interpolate

jpb.mod.interpolate offers two modes of interpolation: stream and event. Stream enables the dynamic interpolation between streaming data points, and event enables the triggering of discrete interpolation events.

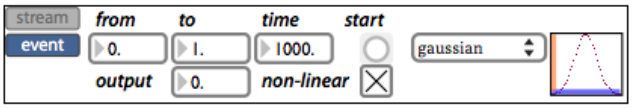

Figure 5: *jpb.mod.interpolate* event function

The *jpb.mod.interpolate* event function extends the Max *line* object by providing more non-linear options, a standard GUI, as well as *pattr* ready variables for saving and recalling of presets.

The stream function extends the *line* object functionality by handling interpolation of variable data streams. One does not have to know the previous value in order to interpolate between values. *jpb.mod.interpolate* handles the interpolation dynamically. The stream function works best with consistent control rates to maintain smooth interpolation.

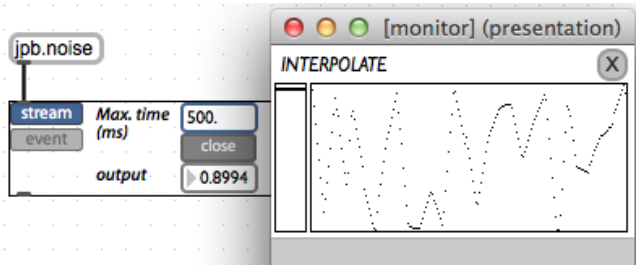

Figure 6: *jpb.mod.interpolate* stream function. Display shows dynamic interpolation of incoming pseudo-random numbers (-1,1).

3.2.2 Thin

jpb.mod.thin removes data by three methods: Sample/Poll, EQ (low pass, high pass), and Event (threshold triggers).

The EQ function filters data using simple high pass and low pass filters, where cutoff is based upon a percentage of the input range (Figure 7).

The event function thins data with thresholds (Figure 8). "min||max" triggers a 1 every time the high threshold is past, and triggers a 0 every time the low threshold is past. "min&&max" is a hysteresis model for triggering

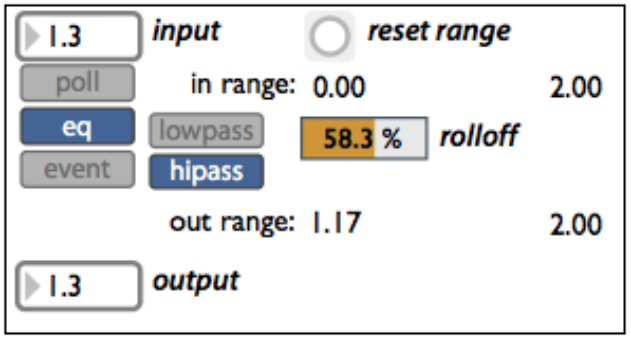

Figure 7: *jpb.mod.thin* eq function, showing the high pass filter.

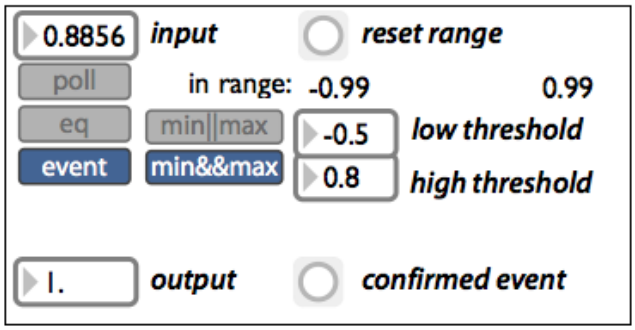

Figure 8: *jpb.mod.thin* event function, showing the hysteresis model.

events, where the maximum threshold triggers **only** after the minimum has been reached.

3.2.3 *Offset*

jpb.mod.offset modifies a number by mathematical addition and subtraction. While simple math problems are more easily executed with basic Max objects, the design process suggested easily selectable common offset equations as a time saving benefit. Thus, the module includes ratio transposition, which parallels the perceptual model of musical transposition as a function of offset.

3.2.4 *Scale*

jpb.mod.scale provides various methods for scaling input data, including non-linear functions. *jpb.mod.scale* moves beyond the Max *scale* object by offering dynamic input range, where high/low inputs values are automatically set by the incoming data stream; single-click output range inversion; four output clipping modes; and seven non-linear functions, all of which the Max *scale* object does not offer.

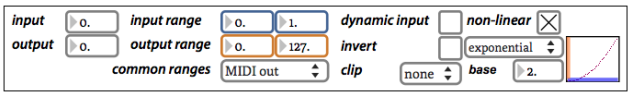

Figure 9: *jpb.mod.scale*, showing 0-1 input to 0-127 output, with an exponential curve applied.

3.2.5 *Smooth*

jpb.mod.smooth provides two methods for smoothing incoming data: step and window. Step averages data over the last X data points. The window method simulates recursive-smoothing techniques commonly found in DSP. The window function takes the average of two data arrays, offset from each other by Y steps, in order to provide a faster transient response.

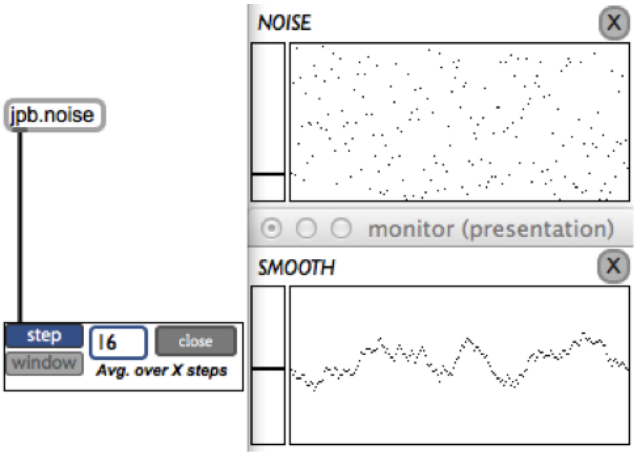

Figure 10: *jpb.mod.smooth*, showing step function.

4. CONCLUSIONS

The *jpb.mod* Max 6 package is a data modification toolkit, built as a means to aid complex mapping development through a rapid prototyping framework. Beginning first with the design need, "Why do I have to make so many connections in order to make my idea work?" *jpb.mod* attempts to provide a solution for creative play and research.

The *jpb.mod* library pushes a rapid prototyping framework by compiling basic data modification tools within easy patchable GUIs. While existing libraries do offer modular design, few offer UI controls, help files, data monitors, and quick-key access. The *jpb.mod* library has already been implemented in eMersion software [27], and used in composing *Triangulation* [28].

This paper presents topics that may elicit further discourse. First, outlining data modification fundamentals provides a platform for addressing mapping problems while attempting to offer language for interdisciplinary collaboration. Second, rapid prototyping (via design thinking) outlines a process toward discovering workable solutions to mapping problems. Third, the development of a Max 6 package offers a rapid prototyping toolkit as proof-of-concept.

Further study into data analysis, data extraction, multi-dimensional data stream modification, and input-output mapping problems would be relevant for creating additional rapid prototyping tools to aid complex mapping projects.

The *jpb.mod* library, along with examples, may be downloaded online at: `http://jpbellona.com/work/jpb-mod`

5. REFERENCES

[1] F. Bevilacqua, R. Muller, and N. Schnell. MnM: a Max/MSP mapping toolbox. In *NIME05 Conference Proceedings*, pages 85–88. New Interfaces for Musical Expression, 2005.

[2] A. Hunt and M. M. Wanderley. Mapping performer parameters to synthesis engines. *Organised Sound*, 7(2):97–108, 2002.

[3] E. R. Miranda. *New Digital Musical Instruments: Control and Interaction Beyond the Keyboard*. A-R Editions, Middletone, WI, 2006.

[4] D. Schwarz, G. Beller, B. Verbrugghe, and S. Britton. Real-time corpus-based concatenative synthesis with

CataRT. In *Proc. of the Int. Conf. on Digital Audio Effects (DAFx-06)*, pages 279–282, 2006.

[5] T. Place and T. Lossius. Jamoma: A modular standard for structuring patches in Max. In *ICMC05 Conference Proceedings*. International Computer Music Conference, 2006.

[6] M. Parker. mp.assignment, a Max input object. `http://www.tinpark.com/2008/12/mpassignment/`, 2009. [Online; accessed 24-Apr-2014].

[7] H. C. Steiner. Towards a catalog and software library of mapping methods. In *NIME06 Conference Proceedings*, pages 106–109. New Interfaces for Musical Expression, 2006.

[8] S. B. Pearce and S. S. Moran. Rapid prototyping in the development of instrumentation for minimal access surgery - a case study. In *First National Conference on Rapid Prototyping and Tooling Research*, pages 29–40, London, England, 1995. Mechanical Engineering Publications.

[9] T. Kelley and D. Kelley. *Creative confidence: unleashing the creative potential within us all*. Crown, New York, NY, 1st edition, 2013.

[10] A. Tanaka. Mapping out instruments, affordances, and mobiles. In *NIME10 Conference Proceedings*, pages 88–93. New Interfaces for Musical Expression, 2010.

[11] J. Ryan. Some remarks on musical instrument design at STEIM. *Contemporary Music Review*, 6(1):3–17, 1991.

[12] D. J. Levitin, S. McAdams, and R. L. Adams. Control parameters for musical instruments: a foundation for new mappings of gesture to sound. *Organised Sound*, 7(2):171–189, 2002.

[13] C. Goudeseune. Interpolated mappings for musical instruments. *Organised Sound*, 7(2):85–96, 2002.

[14] B. Bongers and Sensorband. An Interview with Sensorband. *Computer Music Journal*, 22(1):13–24, 1998.

[15] M. Donnarumma. Xth Sense: a study of muscle sounds for an experimental paradigm of musical performance. In *NIME11 Conference Proceedings*. New Interfaces for Musical Expression, 2011.

[16] C. Udell and J. P. Sain. eMersion: Sensor-controlled Electronic Music Modules and Digital Data Workstation. In *NIME14 Conference Proceedings*. New Interfaces for Musical Expression, 2014.

[17] W. Brent. DILib: Control Data Parsing for Digital Musical Instrument Design. In *Proc. of the 4th Pure Data Convention*, 2011.

[18] M. Donnarumma, B. Caramiaux, and A. Tanaka. Muscular Interactions. In *NIME13 Conference Proceedings*. New Interfaces for Musical Expression, 2013.

[19] D. Watson. *Contouring: A Guide to the Analysis and Display of Spatial Data*. Pergamon, New York, NY, 1st edition, 1992.

[20] J. H. McClellan, R. W. Schafer, and M. A. Yoder. *Signal Processing First*. Prentice Hall, Upper Saddle River, NJ, 1st edition, 2003.

[21] J. Malloch. Digital Orchestra Toolbox. `http://idmil.org/software/digital_orchestra_toolbox`, 2006. [Online; accessed 29-Apr-2014].

[22] B. Benward and M. N. Saker. *Music in theory and practice*. McGraw-Hill, Boston, MA, 2003.

[23] J. S. Simonoff. *Smoothing Methods in Statistics*. Springer, New York, NY, corrected edition edition, 1998.

[24] G. A. Einicke. *Smoothing, Filtering and Prediction-Estimating the past, present and future*. InTech, New York, NY, 2012.

[25] J. Bellona. Kinect Applications (simpleKinect, Kincect-Via-). `http://jpbellona.com/kinect`, 2012. [Online; accessed 01-Jan-2015].

[26] W. Lidwell, K. Holden, and J. Butler. *Universal Principles of Design*. Rockport Publishers, Gloucester, MA, 2003.

[27] C. Udell. eMersion Software. `http://www.unleashemotion.com/store/emotion-software`, 2010-2014. [Online; accessed 14-Dec-2014].

[28] J. Bellona. Triangulation. `http://jpbellona.com/music/`, 2014. [Online; accessed 01-Jan-2015].