DEVELOPING TOOLS FOR STUDYING MUSICAL GESTURES WITHIN THE MAX/MSP/JITTER ENVIRONMENT

Alexander Refsum Jensenius, Rolf Inge Godøy
Department of Musicology
University of Oslo
{a.r.jensenius, r.i.godoy}@imv.uio.no

Marcelo M. Wanderley
Music Technology, Faculty of Music
McGill University
mwanderley@music.mcgill.ca

ABSTRACT

We present the *Musical Gestures Toolbox*, a collection of Max/MSP/Jitter modules to help in qualitative and quantitative analysis of musical gestures. Examples are shown of how the toolbox is used for studying musical mimicry, such as "air piano" performance, and expressive gestures of musicians.

1. INTRODUCTION

Advancements in digital musical instrument design are in part dependent on an improved understanding of how we associate sound-features with human movement [9]. Evidence shows that novices and experts alike tend to spontaneously associate sound-features with specific gestures, although the level of expertise in these associations vary with the degree of musical and/or movement-oriented training. Such spontaneous gesture-sound associations (not necessarily musical) are the results of massive, life-long experience, and may be a valuable source of competence that can be actively exploited in digital audio applications. For this reason we have embarked on a series of studies of music-related gestures [1] , with the aim of extracting a repertoire of gesture-sound associations which in turn may be used as the basis for mapping strategies in digital musical instruments.

In [4] we report on a pilot study of musical "mimicry", i.e. imitation of sound-producing gestures, where subjects ranging from musical novices to experts were asked to play "air piano" while listening to excerpts of piano music . Video recordings from the sessions formed the basis for qualitative analysis of relationships between gestures and musical features. The pilot study confirmed our expectation that the level of precision in mimicking sound-producing gestures gradually increases from novices to experts. Looking at the gestures from a general perspective, comparing the level of effort, speed, density and direction of gestures to the sound qualities, we found that even the subjects with no musical or movement-oriented training performed reasonably well. Future studies will investigate this further, looking at different types of mimicry and also free movement to music.

Based on experiences from the pilot study, and the third author's experiences with studying ancillary gestures of clarinetists [12], we felt the need for software that could help in our *qualitative* analysis of video material. More specifically we needed tools for playing various types of video files, possibilities to easily zoom, crop and rotate the image, change playback speed of the files while preserving pitch, and allow for various types of computer vision and audio analysis. It was also important that the software could run in realtime and be so easy to use that all members in our research groups could get started quickly.

We had been using some of the analysis patches in the Eyesweb environment [2], but its Windows-only existence and limited audio capabilities made it less ideal. Instead we ended up using the Max/MSP/Jitter environment [3], which offers a broad range of objects for sound and video analysis/synthesis, and the benefit of creating standalone cross-platform applications.

Starting with only a simple video playback patch, the project has grown into the *Musical Gestures Toolbox* [2] , a collection of patches built into a set of *modules* [3] that greatly simplify working with video and sound analysis, as well as various types of gestural controllers and sensor interfaces in Max. The next sections will present the different modules and show examples of patches developed for studying musical gestures.

2. THE MUSICAL GESTURES TOOLBOX

The main goal of making the *Musical Gestures Toolbox* was to create tools that would help us in quickly developing patches and programs for studying both gestures and sound. Care has been taken to make the tools scalable and flexible so that they can also be used for control purposes within Max/MSP, and also for use with related software for video/sound analysis and annotation (such as Eyesweb [2], Praat [1] , Anvil [7]) and hardware systems (Polhemus, Vicon, and various sensor interfaces).

[1] More information can be found at http://musicalgestures.uio.no and http://www.music.mcgill.ca/~mtech/clarinet/

[2] Patches, modules and cross platform applications can be downloaded from http://musicalgestures.uio.no.

[3] We call them *modules* since they encapsulate a number of useful objects and include a user interface so that they can be loaded within *bpatcher* objects in Max.

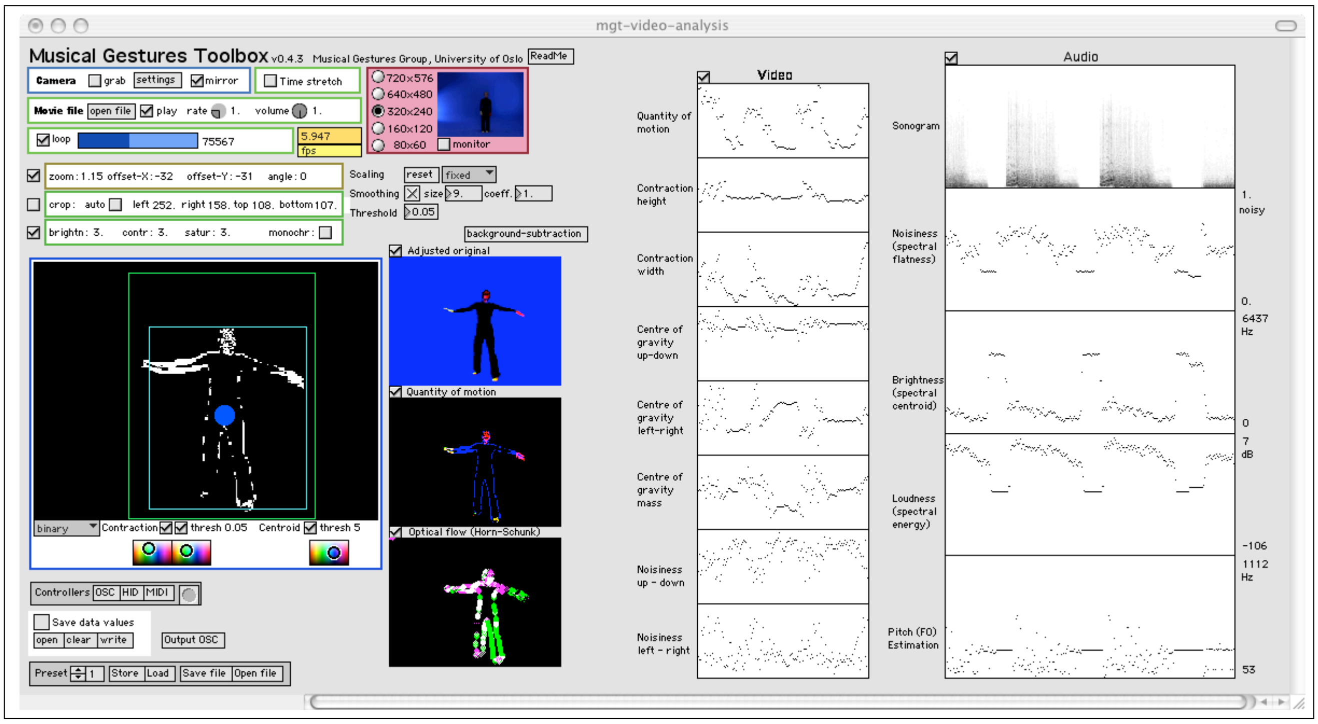


Figure 1. Screenshot of a patch, built with modules from the *Musical Gestures Toolbox*, used for analysis of relationships between music and free dance movements.

2.1. The main modules

The basis for all patches built with the *Musical Gestures Toolbox* are the *source* and *adjustment* modules. They can be seen in the upper left corner of Figure 1. The *source* module includes options for grabbing video directly from a connected DV or web camera, and for playback of any QuickTime-readable video file. Besides easy access to video scrubbing and looping functions, we have also implemented the possibility to change the playback rate while preserving pitch (using the *gizmo˜* pitch shifter available in Max/MSP 4.5). This allows for studying gestures in slow motion, still hearing the original pitch and, to a certain extent, the original timbral qualities. It is similarly useful when analysing different performances where it is necessary to adjust the playback speed to maintain synchronization. Another practical feature is that the playback resolution of the video stream can be changed on the fly, making it possible to work with big patches where adjustments and monitoring can be done at low resolution, before full resolution files are saved in non-realtime.

The *adjustment* module allows for changing brightness, contrast and saturation, as well as controlling zoom, rotation and cropping. In our analytical studies, we find that we often want to zoom in "as much as possible" while still keeping all the person's movements within the frame. To avoid adjusting this manually, which can be tedious when working with long video files, we have implemented an auto-crop function which crops the image based on the maximum contraction values. It is also possible to use the crop-function to focus on a specific part of the image, for example the hand region (see Figure 2).

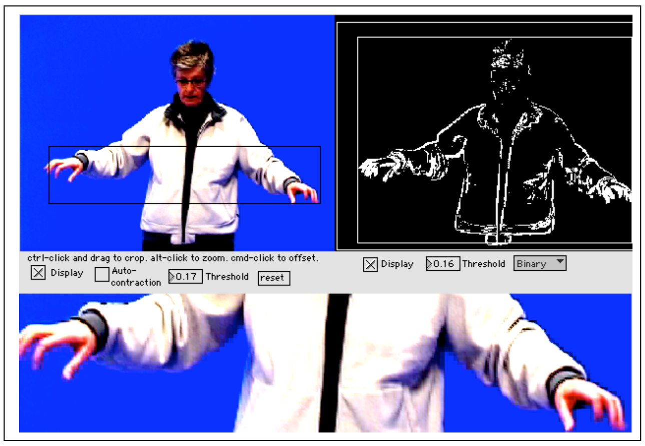
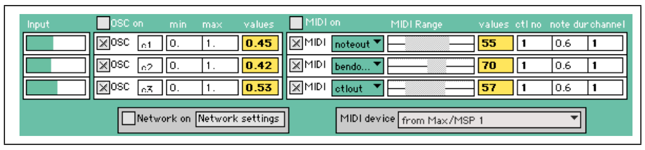

ctrl-click and drag to crop. alt-click to zoom. cmd-click to offset.
Display
Auto-contraction
Threshold
reset
Display
Threshold
Binary

Figure 2. The cropping function in the *adjustment* module makes it easy to zoom and crop the video by clicking and dragging in the source window. Cropping can also be done automatically based on maximum contraction.

The *motion* module can show a number of different video streams (regular, grayscale, quantity of motion, binary quantity of motion, edge detection, inverted or ghost view). It can also display the maximum and running contraction of the person (as bounding boxes), and the centre of gravity in the image. The contraction and centre of gravity can easily be turned on and off, and the colours can be changed to ensure that they are visible on top of the various video streams. For our qualitative analysis, these features are particularly interesting, since they enhance movements that are not so easily seen in the original video. The module also outputs running data values of the



Input	☐ OSC on		min	max	values	☐ MIDI on		MIDI Range	values	ctl no	note dur	channel
	☒ OSC	o1	0.	1.	0.45	☒ MIDI	noteout		55	1	0.6	1
	☒ OSC	o2	0.	1.	0.42	☒ MIDI	bendo...		70	1	0.6	1
	☒ OSC	o3	0.	1.	0.53	☒ MIDI	ctlout		57	1	0.6	1

☐ Network on | Network settings
MIDI device: from Max/MSP 1

Figure 3. Screenshot from the *multicontrol* module where data from the various modules will be automatically detected, scaled to any selected value range and output as Open Sound Control (OSC) messages and/or MIDI.

contraction (height, width, area) and centre of gravity (x-, y-position, mass), which can be graphed with *multislider*, or sent to the *output* module.

2.2. Scalability and flexibility

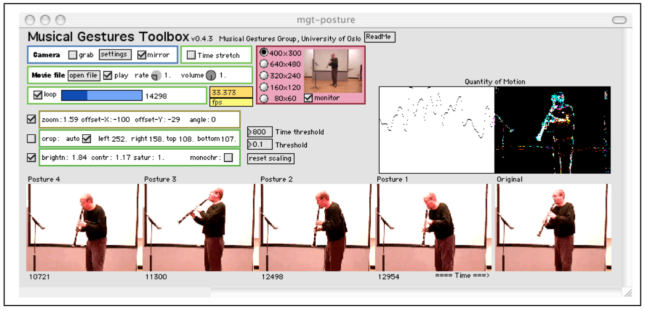
Care has been taken to make the modules as scalable and flexible as possible, so that they can easily be changed and moved without the need to make any other adjustments in the patches. An example of this is how all the modules use the *mgt.scale* object which will automatically scale the values passing the object to floats between 0. and 1, based on the maximum and minimum values of the incoming streams. This works well for most purposes, but in cases where the values vary considerably over time, it might be more useful to turn on the running statistics which will only look at a certain time window. The scaling parameters can be set individually for each module or sent as global values. The same is the case for the *mgt.smooth* object which implements various types of smoothing.

All non-video data are sent as Open Sound Control (OSC) messages [13] between the modules. This makes it easy to save them to a text file, with the *output* module, using the Quicktime timecode as index value. The values can also be used to control external devices with the *multicontrol* module [4] , which will automatically recognize the active data streams, scale them to any selected range and output as OSC-messages on the network and/or to a MIDI-device (Figure 3). We have also used the *multicontrol* module in conjunction with various consumer controllers such as gamepads, joysticks, Wacom tablets and iGesture-pads. Due to the lack of consistency in the value ranges in the input data from such devices, it is of great help to have a tool that can automatically detect the incoming data streams, scale and output them in a consistent manner.

With the *preset* module all parameters can easily be stored and saved to XML-files (using the *pattr* objects available in Max/MSP 4.5).

It goes without saying that working with large video patches consumes a lot of CPU. The modules have been designed to automatically adjust when the resolution of the incoming video stream is changing, so changing the resolution in the *source* module will affect all connected modules accordingly. This makes it possible to work with

[4] The *multicontrol* module is based on the cross-platform application arj.MultiControl [6] resembling Steim's Junxion [11] but with more features and both OSC and MIDI output.



Camera ☐ grab | settings | ☑ mirror
☐ Time stretch
Movie file | open file | ☑ play rate 1. volume 1.
☑ loop 14298
33.373 fps
● 400x300 ○ 640x480 ○ 320x240 ○ 160x120 ○ 80x60 ☑ monitor
☑ zoom: 1.59 offset-X: -100 offset-Y: -29 angle: 0
☐ crop: auto ☑ left 252. right 158. top 108. bottom 107.
☑ brightn: 1.84 contr: 1.17 satur: 1. monochr: ☐
800 Time threshold
0.1 Threshold
reset scaling
10721 11300 12498 12954

Figure 4. Example of a patch, built with the *posture* module, which saves a snapshot every time the change in quantity of motion is bigger than a chosen threshold value. The pictures shift from right to left, and the time code is shown for each captured frame. Here showing postures from a study of expressive gestures in clarinet performance.

unconventional video formats without ending up with output images that are stretched to fit the standard 4:3 format. It also makes it possible to design and test a setup at low resolution, and then save high-resolution files to disk (with the *output* module) for later playback. To further help in improving general performance, all modules are built so that different parts can easily be turned on or off to save processing power.

3. EXAMPLES OF USAGE

The patch in Figure 1 was built to study relationships between gestures and music in dance improvisations. The patch uses the *source*, *adjustment* and *motion* modules, combined with objects for background subtraction and calculation of optical flow from the *cv.jit* collection [10], and sound analysis (brightness, flatness, noisiness and pitch) using the analyzer˜ object [5]. The possibility of having many different visual representations, as well as graphs of both gesture qualities and sound features, is of great help when we study correlations between gestures and sound. Since the graphing capabilities in Max are limited, we also use the *output* module to save data to file for further graphing and analysis in Matlab.

To help in studying salient postures, such as the extremes of movement trajectories, we have made a patch that takes running ”snapshots” of the video stream and shows them next to each other (Figure 4). This is simply based on looking at when the change of quantity of motion goes above a certain threshold, and works quite well after some adjustments of the thresholds. We have also been experimenting with using Hidden Markov Models for recognising postures, and will continue to develop this in future versions [8]. Coupled with running ”trajectories” of the movement, based on different types of delay effects (Figure 5), this helps in understanding how movement changes over time.

The modules have been designed so that they can be used several times in a patch, which makes it easy to quickly build setups for comparative analysis of multiple video

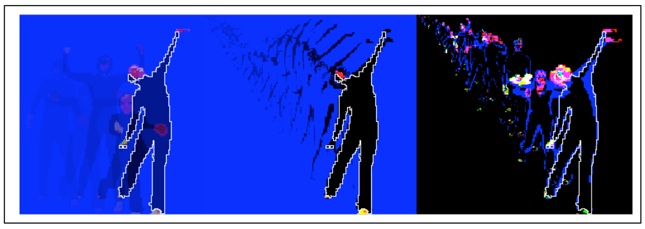


Figure 5. Output of a patch displaying how movements change over time, based on delaying time-spaced frames (left and right) and video feedback (middle).

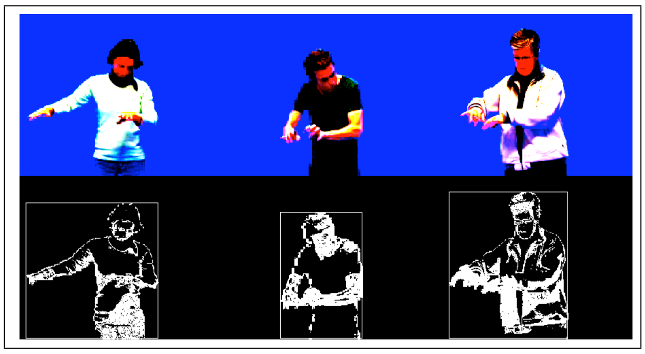


Figure 6. Using modules multiple times makes it possible to build patches for comparative analysis, such as of different air piano performances.

files. Figure 6 shows the output of three different recordings of air piano performances run through the *adjustment* and *motion* modules and then combined to one movie. This works well when the recordings have an absolute time reference, such as in our air piano studies where the audio was identical for each performance. Such comparative analysis is obviously more problematic when the performances are not time synchronized. A system for aligning the videos automatically is one of many things that will need to be developed in future versions.

4. FUTURE WORK

Several issues will be addressed in future development of the *Musical Gestures Toolbox*. Timecoding different types of material is certainly one of the main issues, since all the software and hardware systems we are currently working with seem to do this in different ways (if at all). Synchronising different recordings of the same piece is also necessary, both for comparative analysis and for annotation purposes.

We will also continue to develop tools for high-level analysis of both gestures and sound, and the relationships between the two, and work towards better integration with other systems intended for studying gestures and sound, such as Anvil, Praat and Eyesweb.

5. REFERENCES

[1] P. Boersma and D. Weenink. Praat: Doing phonetics by computer (computer program). http://www.fon.hum.uva.nl/praat/, 8. March 2005.

[2] A. Camurri, M. Ricchetti, and R. Trocca. Eyesweb - toward gesture and affect recognition in dance/music interactive systems. In *IEEE Multimedia Sustems*, Firenze, Italy, 1999.

[3] Cycling'74. Max/msp 4.5, jitter 1.2.3 graphical audio and video environment (computer program). http://www.cycling74.com, 4. March 2005.

[4] R. I. Godøy, E. Haga, and A. R. Jensenius. Playing air instruments: Mimicry of sound-producing gestures by novices and experts. Paper presented at the 6th International Gesture Workshop, Vannes, France, 18-21 May, 2005.

[5] T. Jehan and B. Schoner. An audio-driven perceptually meaningful timbre synthesizer. In *Proceedings of the International Computer Music Conference, Habana, Cuba*, 2001.

[6] A. R. Jensenius. arj.multicontrol (computer program). http://www.arj.no/max/, 1. December 2004.

[7] M. Kipp. Anvil - a generic annotation tool for multimodal dialogue. In *the 7th European Conference on Speech Communication and Technology (Eurospeech)*, pages 1367–1370, Ålborg, 2001.

[8] P. Kolesnik and M. M. Wanderley. Recognition, analysis and performance with expressive conducting gestures. In *Proceedings of the International Computer Music Conference*, Miami, Florida, 2004.

[9] 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.

[10] J.-M. Pelletier. cv.jit computer vision for jitter (computer program). http://www.iamas.ac.jp/ jovan02/cv/, 28. May 2004.

[11] Steim. Junxion (computer program). http://www.steim.org/steim/junxion.html, 8. March 2005.

[12] M. M. Wanderley, B. W. Vines, N. Middleton, C. McKay, and W. Hatch. The musical significance of clarinetists' ancillary gestures: An exploration of the field. *Journal of New Music Research*, 34(1):97–113, 2005.

[13] M. Wright and A. Freed. Open sound control: A new protocol for communicating with sound synthesizers. In *Proceedings of the International Computer Music Conference*, pages 101–104, Thessaloniki, Greece, 1997.