# Harmonic Motion: A Toolkit For Processing Gestural Data For Interactive Sound

Tim Murray-Browne
Queen Mary University of London
Centre for Digital Music
London, UK
t.murraybrowne@qmul.ac.uk

Mark D. Plumbley
Queen Mary University of London
Centre for Digital Music
London, UK
mark.plumbley@qmul.ac.uk

## ABSTRACT

We introduce *Harmonic Motion*, a free open source toolkit for artists, musicians and designers working with gestural data. Extracting musically useful features from captured gesture data can be challenging, with projects often requiring bespoke processing techniques developed through iterations of tweaking equations involving a number of constant values – sometimes referred to as 'magic numbers'. Harmonic Motion provides a robust interface for rapid prototyping of patches to process gestural data and a framework through which approaches may be encapsulated, reused and shared with others. In addition, we describe our design process in which both personal experience and a survey of potential users informed a set of specific goals for the software.

## Keywords

Gesture, mapping, signal processing, software, Kinect.

## 1. INTRODUCTION

Sensing movement and transforming it into sound is at the heart of many interactive performance systems. In particular, the use of contactless sensors such as cameras and the Kinect has become common [6, 7]. The low cost of vision-based sensors and the availability of software to extract features such as an individual's joint positions (also known as skeleton tracking) have led to their widespread use in creating new instruments [9, 2], interactive installations [8], interactive dance [3] and performances blurring the boundaries between dance and instrumental performance [10].

For example, *Hidden Fields*[1] is a dance performance by Danceroom Spectroscopy exploring the physics of invisible energy fields where live visuals and sound are driven by depth information from a rig of calibrated Kinect cameras. An example involving more direct control of sound, *Ethno Tekh* by Brad Hammond and Chris Vik is a dubstep audio-visual performance with one performer on stage controlling seven parameters using a Kinect with skeleton tracking [11].

However, there are a number of factors limiting the accessibility of movement-based sensing to many musicians. Software that interfaces with the hardware can be complex to set up. In addition, interpreting gestural data in a musically meaningful manner is difficult [14]. The challenge presented in creating mappings between sensors and synthesisers that are musical, intuitive, reliable and performative is familiar within NIME research [1, 4]. We currently have a strong body of research identifying frameworks and principles to understand and develop mappings. However, personal experience and discussions with peers has led us to conclude that the difference between success and failure reduces in many cases to iterations of trial and error in manipulating control signals and tweaking 'magic numbers' – constant values chosen because they seem to work, rather than being theoretically derived.

[1] `http://danceroom-spec.com/hidden-fields/`

### 1.1 Example: Expressive dynamics from hand movement

To illustrate our above observation, we present experience from the first author in creating gestural musical controllers as a personal reflection.

*"Impossible Alone is an interactive sound installation I created with the dancer Tiff Chan in 2011. This work included a harmonious synth sound with loudness controlled by the movement of a participant's hand, tracked using a Kinect with the OpenNI library. The aim was to create expressive dynamic shapes that grow in with conductor-like sweeping movements of the hands. However, simply mapping hand velocity to loudness instead produced disjoint and distinctly unmusical results. Over the course of six workshops with Chan, we arrived at a process that produced the intuitive control we had in mind. The process involves several steps of weighted combinations of fixed window moving averages, exponential moving averages and decaying peak signals with five constants arrived at through iterative tweaking.*

*"Later, between November 2012 and August 2013 I conducted a composer residency with the Music Hackspace, a community of musicians, technologists and makers in London devoted to exploring new types of music through creating and modifying technology. Working with a diverse group of seven over this period, including professional musicians and programmers, we created an interactive sound installation exploring musical collaboration, with each member of the group individually creating a new digital musical instrument. Two musicians within the group wished to use the Kinect as a sensor for their instrument. Although they were experienced in creating generative music using Pure-Data and Max respectively, they enlisted my support to develop software to interface with the Kinect and provide an OSC stream of data. Initially, I provided simply the positions and velocities of the joints of the tracked user, but this proved overly complex to process into the interactions they had envisaged. Tasks such as setting up a zone within 3D space that triggers an event or estimating smoothed data requires a knowledge of working with 3D geometry and rudimentary signal processing. In the end, I created two bespoke*

*pieces of software: one to set up and create trigger zones, and one which reproduced the sweeping dynamics control from Impossible Alone described above."*

In these two cases, we can observe that there are a range of technical skills typically involved in receiving and processing sensor data, which can render the technology inaccessible to many. Furthermore, a technique for processing gestural data was developed, not by researching standard references, but iteratively through trial and error. Although bespoke to the project it was created for, a need for a similar technique arose in another project. However, reproducing it took time, and the tacit knowledge of its creator. These two issues formed the initial motivation to create Harmonic Motion.

## 2. REQUIREMENTS ANALYSIS

There are a number of tools currently available to musicians interested in working with gestural data from a device such as the Kinect. OSCeleton[2] and Synapse[3] are open source programs that read depth image and perform skeleton tracking from a Kinect using the OpenNI/NITE library[4]. Likewise, there are external objects available for Max[5] and PureData [12] that perform a similar function, as well as library packs for creative coding environments such as Processing,[6] openFrameworks[7] and Cinder.[8] However, in our experience, extracting meaningful features to create the musically expressive control one has in mind is where the challenge lies.

On the other end of the spectrum is a tool such as the Wekinator [5], which applies supervised machine learning algorithms to map input features to output parameters based on training examples provided by the user. However, although some feature extractors such as video edge detection are included, the system is geared towards developing complex mappings of pre-extracted features.

Harmonic Motion sits somewhere between these two ends of the spectrum.

### 2.1 A survey of potential users

To better understand our potential user base and their requirements, we conducted an online survey. Contacting primarily NIME and creative coding related forums and mailing lists, interested individuals were invited to sign up to a news mailing list for the project. Those who did so were subsequently invited to fill out an online questionnaire, of whom 96 did so. As our aim was to assess requirements for this project, the survey was not conducted at a standard suitable for formal statistical analysis. However, the results are of interest to explain design decisions behind the toolkit.

There was a diverse range of musical backgrounds and intended use for the toolkit, with the most popular being creating new instruments, augmenting existing instruments through controlling effects and creating interactive installations. When asked an open question *'Are there any problems you have faced in previous attempts to work with gestural data?'*, 57% of respondents answered with the issues raised most commonly as designing mappings (20% of respondents), speed and latency (14% of respondents) and hardware-specific problems (10% of respondents). Seven percent of respondents also reported stability problems, with some existing software prone to crash.

Much of the information collected confirmed our existing beliefs. We were, however, surprised to find a high level of interest from those who consider themselves advanced or professional in programming using code (31% of respondents), patch-cord based languages (62% of respondents), as well as those less experienced. 81% considered themselves intermediate or higher in developing mappings between user input and synthesis parameters with 97% being at least familiar with patch-cord based programming. Although our initial goal was to focus on less advanced users, this greater understanding of the range of experiences of potential users led us to broaden the scope of the toolkit to ensure it was a tool powerful enough for advanced users while remaining accessible for those less experienced.

## 3. GOALS

From this survey, combined with our own research and experience, we arrived at the following goals for Harmonic Motion.

**Rapid Prototyping.** The system should allow feature extraction processes from gestural data to be rapidly prototyped through iteration. In particular, it should be easy to tweak parameters and quickly see the effect through visual feedback.

**Usable.** It should be simple and well documented to set up and use the system.

**Reusable.** Processing techniques that have been developed by users should be reusable in other projects with minimal effort, and easily combined with other techniques.

**Shareable.** Users should be able to share processing techniques they have developed, allowing newcomers to get started more quickly

**Extensible.** Without compromising usability for those without coding or maths experience, the system should provide a means for advanced users to create, encapsulate and release their own advanced gestural processing techniques.

**Stable.** Users of the system need confidence that it will not crash or start behaving unpredictably. Not only can a crash ruin a performance, but installation software may be left unattended for long periods of time.

**Fast.** There is often a lot for one computer to do when working with audio. The software should be efficient and make use of multiple processing cores without introducing complexities to users.

**Free.** Dependencies on proprietary or restrictively licensed software should be avoided as far as possible.

**Complementary.** Harmonic Motion is not intended to replace existing tools such as Wekinator or Max, and should be adaptable to complement them.

Additionally, although Harmonic Motion is being created primarily for individuals working with gestural control of sound, it may also be of use to other artists working with gestural interaction and should be open to wider use cases.

[2]`http://github.com/Sensebloom/OSCeleton`
[3]`http://synapsekinect.tumblr.com/`
[4]`http://structure.io/openni`. At the time of writing (April 2014), the NITE library which handles skeleton tracking on a Mac is no longer publicly available. We are currently considering alternative solutions for Mac users looking for skeleton tracking.
[5]`http://cycling74.com/products/max/`
[6]`http://processing.org`
[7]`http://openframeworks.cc`
[8]`http://libcinder.org`

## 4. THE TOOLKIT

Some of our above goals required consideration at the outset of the project. Introducing dependencies on third party frameworks can bring licensing restrictions that our users would then become bound by. Different development languages have different advantages in terms of runtime speed and the ease of which they support user-written extensions without compromising stability.

We chose to develop in C++ due to its speed, the established availability of hard-specific SDKs such as the Kinect. After a review of features, stability and licensing requirements, we chose to use third party libraries Cinder, OpenCV[9] and Boost[10], with the graphical interface further making use of the Qt[11] framework.

Due to the familiarity with paradigm of patch-cord based programming languages such as Max, PureData or Touch Designer indicated by our survey, gestural processes are created and visualised as a network of interconnected nodes, although with a number of key differences to achieve our goals.

### 4.1 Nodes and network

We describe the overall data process being designed by the user as a *patch*. Similar to Max or PureData, a patch is made up of a directional network of processing nodes, with the user connecting the outputs of nodes to the input of other nodes. However, rather than existing as minimal blocks as is the case in Max or PureData, nodes are intended to be able to encompass entire techniques in themselves, with their own sets of parameters and drawing functions. Included with the library are nodes for using the Kinect (versions 1 and 2) and LEAP Motion sensor, with plans to develop support for other controllers in the future.

There are a range of datatypes accepted as input and output currently implemented, focusing primarily on working with image and pose data: real-valued number, 2D/3D vector, 3D skeleton of joint vectors, 2D skeleton (joint vectors as projected onto a depth image), 2D/3D scene (a collection of skeletons in the same coordinate space), 2D/3D image. We describe the value of an above datatypes from a specific point in time as a *frame*. Data may be broadcast from Harmonic Motion using OSC or MIDI, or through the C++ API described below.

To facilitate rapid prototyping and making the software easy to use, patches are created and developed in a graphical interface. This provides real-time visual feedback of how the data is being transformed as it passes through the patch, and quick access to parameters to tweak this.

### 4.2 Parallel processing

Patches are designed to allow individual nodes requiring a heavy amount of processing to run on a separate thread. To allow this, data travels through the patch asynchronously: while a threaded node is processing one frame of data, another may arrive and be queued for it to process. This contrasts with Max or PureData where the entire network is traversed by one frame before the next is processed.

This approach has the advantage of allowing a patch to implement a lengthy process without dropping the framerate at which data is processed (although length processing will still introduce latency). However, where a single input frame is being processed to produce multiple outputs through different pathways in the patch, the outputs may arrive at different times, which may be undesirable to the end user. We are presently exploring methods of resynchronising frames as they pass through the system using timestamps.

### 4.3 Modularity

To allow processing techniques to be easily reused and shared and combined with each other, patches can be exported and imported to file in the human-readable JSON format. Exporting a patch not only encapsulates the network of nodes that has been created, but the values of any parameters registered by those nodes.

### 4.4 C++ API

Harmonic Motion may be used entirely through its graphical interface, sending OSC or MIDI output. However, it has also been designed to allow it to be integrated within an external C++ project. In particular, users working in C++ may prototype and develop a patch using the patch-based user interface and then import and run this patch within a C++ project without the need to simultaneously run separate software. We hope in the future to be able to provide external objects to allow similarly integrated behaviour for Max and PureData.

To ease integration of the C++ library, we have sought to reduce external dependencies in creating the library, avoiding restrictively licensed software (where possible – this is discussed further in Section 5.1, and plan to provide addon packs for the creative coding environments openFrameworks and Cinder, which are both in C++.

### 4.5 Node SDK

Our goal of extensibility is to allow advanced users to implement their own nodes using a lightweight API. As threaded processing of a shared data source can be complex and prone to bugs that arrive unpredictably, the API has been designed to shield developers from threading aspects of the software. At present, the build process requires the user to rebuild the entire toolkit in order to integrate a new node. We are exploring approaches to allow custom nodes to be integrated into the software without this rebuild. This will likely be realised either through creating a system to build nodes into a dynamic library format, or through integrating an interpreted language such as Lua for custom nodes to be built with.

Our intention is that the SDK will allow new gestural signal processing approaches (e.g. Skogstad et al.'s [13] custom filters for real-time motion capture) to be easily released and put into use by artists and musicians.

## 5. DISCUSSION

It is our belief that creative toolkits are often released as a personal project, encapsulating software individuals have created for their own practice, making it more widely available and then occasionally improved and adapted based on feedback from potential users. Harmonic Motion follows this trend, however we have also sought to adopt a methodical assessment of our potential users' requirements during the design stage.

Our first four goals of allowing patches to be rapidly prototyped, reused and shared in an easy to use environment, are addressed through adopting a graphical patch-cord system, which our survey found to be at least familiar to 97% of our potential users. Creating patches involves not only connecting nodes, but tweaking a large number of parameters to achieve the behaviour desired. Our graphical interface has been designed from the outset to make it easy to adjust parameters and instantly see the effects, not only on the output but on the data travelling between nodes. Reuse

[9]http://opencv.org
[10]http://boost.org
[11]http://qt-project.org/

and sharing of patches is achieved through functionality to import and export to files. Again, parameter values are fundamental, and these are included when doing so.

Our goal of remaining complementary to other software led to the decision to separate the user interface from a core library. In addition, we provide customisable OSC output, allowing users to output messages directly to Wekinator, or in the same format as OSCeleton, for example.

The core of the library has been designed so that requirements of stability and speed are balanced with extensibility and the ability for rapid prototyping. Both parallelism and the capability of nodes to be created and destroyed at any point during runtime introduce potential stability risks arising through concurrent data access and incorrect or absent memory deallocation. The potential for problems increases with code introduced by users who may be unaware of what mechanisms have been implemented to guard against these risks. However, the use of a node-based design has allowed us to encapsulate and hide the process by which data is shared between nodes, providing users with a simple and safe basis to work from.

## 5.1 Licensing

Finally, licensing is important to consider from the outset of creating software as introducing external libraries can limit the licence under which our software may be released. Although we are still considering the exact licence under which to release Harmonic Motion, we believe it is important to reduce any barriers of use introduced by a licence as far as possible, including in commercial and closed source projects. The essential third-party dependencies are:

**Cinder,** released under the MIT licence,

**Boost,** released under its own Boost licence,

**OpenCV,** released under the 3-clause BSD licence,

**Qt,** released under the LGPL licence, not necessary when using the C++ API.

All are permissive licences that allow commercial use in closed source applications (LGPL requires Qt to be dynamically linked but this is acceptable in our application). Other proprietary libraries that are necessary for certain use cases, such as the Microsoft SDK for the Kinect and the LEAP Motion SDK carry their own requirements and limitations, which some users may wish to avoid. We are exploring ways in which these components, and hence their licences, may be optionally excluded from the toolkit.

## 6. CONCLUSION

Harmonic Motion provides an accessible and powerful means for musicians, artists and designers to create bespoke yet reusable approaches to making sense of gestural sensor data. As a piece of software, it allows users to see instantly the effects of how they are processing and extracting features from data, rapidly develop and refine these techniques and retain their work for future projects.

With the ongoing evolution of sensor technology and operating systems, a key challenge in creating an open source toolkit such as Harmonic Motion is to establish a community of contributors to assist in its maintenance. Based on responses to the survey described above, we are optimistic that we will be able to achieve this. Any readers interested in being involved are encouraged to get in touch.

## 7. ACKNOWLEDGMENTS

Harmonic Motion is funded by the UK Engineering and Physical Sciences Research Council (EPSRC) under the Platform Grant (EP/K009559/1). We are grateful for the insightful feedback received from our reviewers.

## References

[1] J. Drummond. Understanding interactive systems. *Organised Sound*, 14(2):124–133, 2009.

[2] X. Fan and G. Essl. Air Violin: A body-centric style musical instrument. In *Proceedings of the International Conference on New Interfaces for Musical Expression*, pages 122–123, Daejeon, Korea, 2013.

[3] S. Fdili Alaoui, C. Jacquemin, and F. Bevilacqua. Chiselling bodies augmented dance performance. In *Proceedings of Computer Human Interaction (CHI) (extended abstracts)*, pages 2915–2918, Paris, France, 2013.

[4] S. Fels, A. Gadd, and A. Mulder. Mapping transparency through metaphor: Towards more expressive musical instruments. *Organised Sound*, 7(2):109–126, 2002.

[5] R. Fiebrink, D. Trueman, and P. R. Cook. A meta-instrument for interactive, on-the-fly machine learning. In *Proceedings of the conference on New Interfaces for Musical Expression*, Pittsburg, PA, 2009.

[6] W. Fohl and M. Nogalski. A gesture control interface for a wave field synthesis system. In *Proceedings of the International Conference on New Interfaces for Musical Expression*, pages 341–346, Daejeon, Korea, 2013.

[7] A. L. Fuhrmann, J. Kretz, and P. Burwik. Multi sensor tracking for live sound transformation. In *Proceedings of the International Conference on New Interfaces for Musical Expression*, pages 358–362, Daejeon, Korea, 2013.

[8] C. Honigman, A. Walton, and A. Kapur. The third room: A 3d virtual music paradigm. In *Proceedings of the International Conference on New Interfaces for Musical Expression*, pages 29–34, Daejeon, Korea, 2013.

[9] A. R. Jensenius. Kinectofon: Performing with shapes in planes. In *Proceedings of the International Conference on New Interfaces for Musical Expression*, pages 196–197, Daejeon, Korea, 2013.

[10] A. Johnston. Fluid simulation as full body audio-visual instrument. In *Proceedings of the International Conference on New Interfaces for Musical Expression*, pages 132–135, Daejeon, Korea, 2013.

[11] P. Kirn. Full-body music, live: AV Kinect performance, complete with beer bottle. *Create Digital Music*, 12 November 2012. `http://createdigitalmusic.com/2012/11/full-body-music-live-av-kinect-performance-complete-with-beer-bottle-video/`.

[12] M. Puckette. Pure Data: Another integrated computer music environment. In *Proceedings of the Second Intercollege Computer Music Concerts*, pages 37–41, Tachikawa, Japan, 1996.

[13] S. A. Skogstad, K. Nymoen, M. Hovin, S. Holm, and A. R. Jensenius. Filtering motion capture data for real-time applications. In *Proceedings of the International Conference on New Interfaces for Musical Expression*, pages 143–147, Daejeon, Korea, 2013.

[14] E. Zhang and R. Fiebrink. Kib: Simplifying gestural instrument creation using widgets. In *The International Conference on New Interfaces for Musical Expression (NIME05), Vancouver, Canada*, pages 519–524, Daejeon, Korea, 2013.