Non-invasive sensing and gesture control for pitched percussion hyper-instruments using the Kinect

Shawn Trail, Michael Dean, Tiago F. Tavares
Dept. of Computer Science
University of Victoria
Victoria, Canada
shawn@intrinsicaudiovisual.com

Gabrielle Odowichuk
Dept. of Electrical and Computer Engineering
University of Victoria
Victoria, Canada
godowichuk@gmail.com

Peter Driessen
Dept. of Electrical and Computer Engineering
University of Victoria
Victoria, Canada
peter@ece.uvic.ca

W. Andrew Schloss
School of Music
University of Victoria
Victoria, Canada
aschloss@uvic.ca

George Tzanetakis
Dept. of Computer Science
University of Victoria
Victoria, Canada
gtzan@cs.uvic.ca

ABSTRACT

Hyper-instruments extend traditional acoustic instruments with sensing technologies that capture digitally subtle and sophisticated aspects of human performance. They leverage the long training and skills of performers while simultaneously providing rich possibilities for digital control. Many existing hyper-instruments suffer from being one of a kind instruments that require invasive modifications to the underlying acoustic instrument. In this paper we focus on the pitched percussion family and describe a non-invasive sensing approach for extending them to hyper-instruments. Our primary concern is to retain the technical integrity of the acoustic instrument and sound production methods while being able to intuitively interface the computer. This is accomplished by utilizing the Kinect sensor to track the position of the mallets without any modification to the instrument which enables easy and cheap replication of the proposed hyper-instrument extensions. In addition we describe two approaches to higher-level gesture control that remove the need for additional control devices such as foot pedals and fader boxes that are frequently used in electro-acoustic performance. This gesture control integrates more organically with the natural flow of playing the instrument providing user selectable control over filter parameters, synthesis, sampling, sequencing, and improvisation using a commercially available low-cost sensing apparatus.

1. INTRODUCTION

Hyperinstruments are augmented acoustic instruments with added electronic sensors used as gesture acquisition devices. Designed for interfacing a computer in a more intuitively musical nature than conventional means, they leverage the performer's expertise [**?**]. The pitched percussion family is problematic when included in electro-acoustic contexts because it is difficult to acquire a direct signal for sound reinforcement, signal processing or generating control data. While typically used for audio reinforcement, dynamic and condenser microphones are prone to feedback, don't fully capture the signal and pick up unwanted environmental artifacts. These are poor conditions in which to generate control data from. It is also desirable to avoid directly adding electronics and sensors to the instrument. A common approach to providing digital control without modifying the instrument is the use of an external interface such as a fader/knob box or a set of foot pedals. However, this is problematic as the performer has to stop playing in order to interface the computer. Foot pedals are a little bit better but they can still be distracting.

To address these concerns we have developed a set of tools that take advantage of dynamic gestural parameters that are intrinsic to pitched percussion performance. Our tools are designed for capturing these gestures in musically meaningful, non-invasive ways in order to control the computer in performance. They are based on a combination of non-invasive sensing using the Kinect and high-level gesture control detection. The goal is to develop novel tools that any performer can easily adapt and use without substantial technical installation requirements or musical restraints. Our system also addresses cost issues which often make music technology practice prohibitive. We have applied these ideas in the vibraphone, the Gyil (an african xylophone from Ghana), and the Likembe a lamellophone from central africa. In this paper the vibraphone is used as a canonical example.

2. PREVIOUS WORK

This paper builds on previous work in the field of percussion based gesture sensing, machine learning, and machine musicianship. Work by Overholt [**?**] and Kapur [**?**] have greatly influenced the development of the interface solutions presented. We also look at Machine Musicianship in which a computer systems can analyze and perform musical events autonomously [**?**] and have also been influenced by Shimon, a robotic marimba [**?**].

Commercial mallet-percussion based MIDI control solutions are limited. The Simmons Silicon Mallet and MalletKat both incorporate rubber pads with internal FSR's laid out in a chromatic keyboard fashion and vary in octave range. This Simmons offered very little configurability as it was basically an autonomous instrument and had a 3 octave mallet keyboard controller played with either vibraphone mallets or drum sticks. The Malletkat offers the same typical internal MIDI configurability as high-end pi-

NIME'12, May 21 – 23, 2012, University of Michigan, Ann Arbor.

ano style keyboard controllers and can connect to any MIDI sound module. K&K Sound[1] produced a (nowadays discontinued) piece of hardware that extracts control data from a vibraphone using an array of sensors. The Xylosynth, by Wernick[2], is a digital instrument that aims at providing the same haptic response of a wooden xylophone while yielding MIDI instead of acoustic data. The Marimba Lumina is another mallet style controller, and is able to gather velocity, position and contact. Additionally, it allows one to relate different controls to be triggered when playing different bars. These instruments are, in general, technologically robust, but unable to offer the same haptic feedback as an acoustic instrument and typically require the instrumentalist to modify playing technique in order to achieve successful results. They are also costly.

Our work has been influenced by several new music instrument ideas beyond the pitched percussion family. The Theremin is unique and relevant in that it is played without physical contact controlled by 3D hand gestures. The modern Moog Ethervox, while functionally still a theremin, can also be used as a MIDI controller, and as such allows the artist to control any synthesizer with it. With the development of new gaming interfaces based on motion controls, it became easier to generate musical interfaces controlled by movements. That allows one to use natural motion, which, as observed by [**?**], is an inherent part of the performance of a pitched-percussion player. Motion-based game controllers were used as musical tools in [**?**], taking as basis the WiiMote device. In [**?**], the idea of using the Kinect as a controller to musical expression is discussed. In that work, the Kinect was used as a low cost way to obtain position data in order to enhance the soundscape manipulation capabilities of a performer.

3. NON-INVASIVE SENSING

In our previous work involving non-invasive sensing [**?**], the Kinect and it's associated software libraries were used to perform human skeleton tracking. This tracking is used to produce estimated positions of the performers limbs, which can then be mapped to musical parameters. Each cartesian axis of motion was maped to filter parameters that modify live and sampled audio from the vibraphone, creating a control volume accessed through the mallets in the space over the keys. A new extension of this work introduces a form of augmented reality to our hyper-vibraphone. Using the Kinect webcamera and computer vision libraries, we are able to detect the position of the percussionist's mallet tips. This tracking is used to augment each bar of the vibraphone with the functionality of a fader. Using this technique on a 37 bar vibraphone, it is possible to provide the functionality of 37 virtual faders that may be used as traditional controllers. This augmentation, illustrated in Figure 1, provides an intuitive platform that allows the performer to control a large number of sliders without turning away from the instrument.

Currently we are tracking mallet tips based on their color. In order to detect the positions of the mallet tips, the color image from the video camera is transformed into Hue, Saturation, and intensity Values (HSV). Each of these signals is thresholded to filter out unwanted colors. The resulting signals are combined, and a contour detection algorithm is executed. This process yields bounding rectangles that identify the mallet tips. The centroid of the bounding rectangle is assumed to be the position of the mallets in the virtual representation.

[1] `http://www.kksound.com/vibraphone.html`
[2] `http://www.wernick.net/`

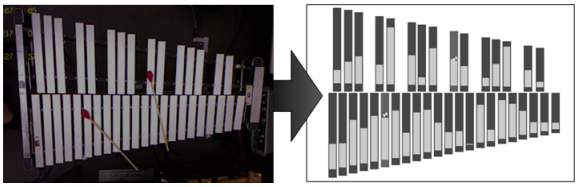

Figure 1: Virtual Vibraphone Faders

Determining the position of the mallet tips in terms of the vibraphone bars requires the creation of a virtual representation of the instrument. This representation was created using explicit measurements of the vibraphone's bars. These measurements were used to create a model consisting of the set of bars, each with a corresponding placement, size, pitch, and an associated control value. The algorithm supplies the mallet positions relative to the camera, but we actually want our position data in the virtual space. Effectively mapping the tracked mallets involved several transformations, and requires a calibration phase in which the position of the vibraphone with respect to the camera is also recorded. Once we have obtained the position of the mallets in the same space as the vibraphone, the system yields information on what bar is currently covered by the mallet, and a fader value associated with that bar. A delay-based activation time was added to the algorithm, so that the performer must pause on each bar for a pre-defined time before the sliders will start to change.

The 640x480 resolution used in our prototype is sufficient to perform accurate detection of the mallet positions. The main resolution restriction is that the camera should point at the vibraphone from a distance that makes each bar to be present in a reasonable number of pixels. While this may suggest using a higher resolution, it is important to note that the algorithms must be executed in real-time, therefore less data is desirable. Future work will involve fusing the video camera data with IR sensors data, as well as using motion tracking algorithms in combination with our current contour detection algorithms. This fusion will improve the robustness of the tracking system, and address the current sensitivity to changes in ambient lighting.

4. FORECASTING OF POSITION DATA

We describe a method that is capable of forecasting the continuation of a given data array $x[n]$, where $x[n]$ is the value of the n-th reading of a certain sensor. The forecasting algorithm does not require any previous training on previous templates, which means that the musician has freedom to improvise and generate creative soundscapes and define the appropriate gestures while performing. The key idea is that the gesture is identified by being repeated without requiring a preset looping duration. When forecasting, the system provides data which aims to be equivalent to the sensor data that would be yielded if the musician continued the previous gesture. This allows the performer to continue playing while still getting the soundscape variations derived from sensor data. The forecasting method is based on evaluating what time lag, in samples, is most likely to be the fundamental period of the received data. This means that although the motion may be freely composed and improvised only repetitive motions can be predicted by the system. The method begins by estimating the autocorrelation of the last N input

samples, which is defined as:

$$r_x[k] = \sum_{n=0}^{\frac{N}{2}-1} x[n]x[n-k]. \quad (1)$$

An autocorrelation signal presents peaks at positions k that correspond to the time lags to which $x[n]$ is mostly self-similar. However, there are some other peaks, especially at the zero time lag $k = 0$ that must be filtered out. In order to do that, a signal $r'_x[k]$ is calculated by upsampling $r_x[k]$ by a factor of two and subtracting it from the original signal. The value $j = \arg\max r'_x[k]$ is obtained by a simple linear search. The forecasting, then, proceeds by yielding an estimate $\hat{x}[N+1] = x[N-j+1]$. The quality of the estimate may be evaluated by the ratio $r_x[j]/r_x[0]$, which will present values closer to 1 when the signal is significantly periodic and values. The forecasting system works in three different modes: *learning*, *predicting* and *bypass*. The *learning* mode should be triggered when a new gesture is to be acquired. While in this mode, the internal buffer is updated and the forecasting algorithm is executed at each new sample. Since *learning* mode assumes that the gesture is not yet learned, the system yields bypassed data instead of predicted data.

When the *prediction* mode is triggered, the system starts yielding forecasts, adding them to the internal buffer as if they were received as inputs. While in this mode, the system does not recalculate the forecasting algorithm, sticking with a single value for j during the whole process. The operation of *bypass* mode is to simply bypass input data to the output, while ignoring any operations regarding processing. Hence, if the *prediction* mode is triggered while the system is in *bypass* mode, the system will recall the last learned pattern. When operating in the *bypass* mode, the system preserves the last learned gesture, hence it may be used again by triggering the *prediction* mode.

The internal operation of the system is visualized in Figure 2, which shows the input (bypassed) data from a one-dimensional sensor and the predicted data. The figure was generated while acquiring manually-driven data from a one-dimensional sensor. In the first five seconds, the system is in bypass mode, and the prediction system has received any samples yet. When the learning mode is triggered, the predicted data quickly synchronizes with the repetitive sensor data received as input. When the prediction mode is triggered, the forecasting system continues the learned gesture and ignores the sensor data.

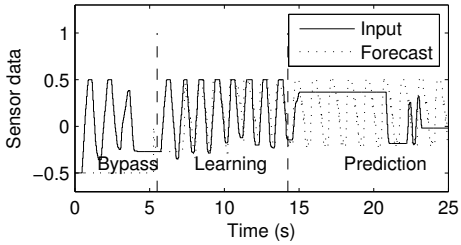

Figure 2: Visualization of different modes of operation for the sample forecasting system.

5. RECOGNITION OF DRUM PATTERNS

Another approach to gesture control is to recognize percussive patterns from a user defined set of patterns in order to initiate computer-controlled musical events. The proposed system aims at providing the performer the possibility of influencing the computer ensemble using musical cues without having to alter the playing technique, which represents a significant improvement over conventional external tactile controls. Our approach is based on calculating the similarity between an input drum pattern, played at a certain point during the execution of a piece, and a previously recorded pattern. The similarity is a continuous value that is greater when the pattern played is closer to the recorded pattern. If the similarity value exceeds a certain user-defined threshold, the system triggers an associated action.

Two equal length sequences of numerical data can be compared in terms of a single value between one and zero. A value of one indicates the two sequences are identical, whereas zero indicates a drastic lack of similarity between the two data sets. This similarity computation is known as correlation. Equation 2 computes a value that describes similarity between two sequences.

$$\text{Similarity} = \frac{\sum_{n=0}^{N-1} x[n]y[n]}{\sum_{n=0}^{N-1} x[n]^2 \times \sum_{n=0}^{N-1} y[n]^2} \quad (2)$$

To account for slight variation, sequences can be convolved with a Gaussian function with standard deviation σ and mean μ, as shown in Equation 3, before being compared. This process results in a smearing of the sequence's values to adjacent locations, as seen in Figure 3.

$$f(x) = \frac{1}{\sigma\sqrt{2\pi}} e^{\frac{(x-\mu)^2}{2\sigma^2}}. \quad (3)$$

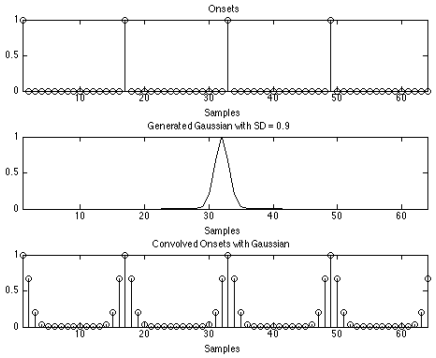

Figure 3: Convolution with Gaussian function.

To assemble such sequences in real time, the proposed system assumes that the player will perform to a specific tempo. For each time interval in the measure, if an onset is detected, it is notated in the sequence using a value of one. Otherwise a value of zero is used indicating a rest. The onset algorithm used calculates the power of the input signal and triggers onsets when a threshold has been surpassed [**?**].

From the methods described, similarity between musical patterns can be calculated. To do so, the system must have two modes of operation: recording a template pattern and comparing incoming patterns to the template pattern. When comparing incoming patterns the similar value of the input pattern to a set of stored templates is calculated and if it exceeds a user defined threshold an event is triggered.

In order to gain perspective on the effectiveness of the system in a real performance situation, the pattern recognition system was tested with a human percussionist performing on a traditional West African pitched percussion instrument known as the Gyil. The instrument provided audio input to the software through the use of contact microphones that were mounted to each bar of the instrument. The pattern recognition system was used to recognize patterns played on

a single bar. Of the two types of tests done, the first provided similarity values when a pattern was played during a performance. In this first test, the sampled bar was only played when the user wished to trigger the software using a pattern. Otherwise, the bar was avoided as the performer played the instrument. In the second test, the entire instrument was utilized by the performer; the sampled bar was used for both triggering patterns and for playing.

The first test used five unique, one measure long rhythms that were tested ten times each at two different tempos. The two tempos used to test varied by one-hundred beats per minute. For all one-hundred measurements in the first test, an average similarity of 0.731 was calculated. The measured similarity values were affected most drastically by two factors: the human inaccuracies when recording the template pattern and the human inaccuracies when repeating the pattern in order to trigger some action. The second test used a fixed threshold of 0.731 at 172 beats per minute (BPM) in order to test how often the system incorrectly triggered an action during a sixteen bar solo. During each solo performance, the user played rhythms on the sampled instrument bar every second measure. The number of times an action was not triggered when the player intended to do so was also documented. Of the ten tests for these sixteen bar solos, the system incorrectly identified a player's pattern as being the template pattern only twice, or 0.025% of the measures in which the bar was played. Of these two incorrectly identified patterns, there was stark similarity to the template pattern, and it was unsurprising that the system gave a false positive result. When the player intended on triggering an action by performing the template pattern, these tests gave no negative results.

6. DISCUSSION

All systems described in this paper were implemented with the aim of being used in a computer music environments such as Max/MSP[3] and Ableton Live[4]. The computer vision system uses the openCV [5] and openFrameworks [6] libraries. It is compiled as a standalone program that yields Open Sound Control (OSC) messages. The gesture repetition algorithm described in Section 4 was implemented as a patch for Max/MSP. This allows the user to design the exact means by which each mode will be triggered – the user may require a specific button to trigger each mode, or simply a button that cycles through all modes, or even some other interface that makes more sense in an specific context. The patch also offers as feedback the current value for the maximum autocorrelation coefficient and the index of that value – when both remain more or less constant for some time, it means that the system has acquired a certain gesture. Last, the drum pattern detection algorithm described in Section 5 was implemented using the Max4Live[7] framework.

This paper has presented different ways of augmenting pithed percussion instruments such as the vibraphone with digital control capabilities. Using our approach they can be turned into hyper-instruments without requiring invasive additions to the instrument. One of the authors is a skilled vibraphone instrumentalist with a background in computer music. From the perspective of a skilled vibraphone performer, this toolset works as it should and offers a level of interaction with the computer previously unavailable without interrupting native performance techniques (setting the mallets down to interface tactile controllers, etc). The virtual faders are responsive and accurate to the point that using the mallets tips on the surface of the bar proves a sufficient and intuitive fader style interface. The technique required is similar to the pitch bend extended technique. The system could be improved by adding vertical detection of the mallet tip to engage/disengage rather than only a time based approach. This would solve the problem of accidental activation when playing on a single bar long enough to engage it's respective fader. The two methods of gesture control based on repetition and rhythmic patterns offer effective ways to transmit control data while holding the mallets. Issues with calibration and mappings could be assisted with a more comprehensive GUI. While the system can stand to be improved, it is ready, as is, to be used within its known limitations on stage in performance. In future work, further improvements on the usability and effectiveness of the proposed algorithms will be implemented. Media files related to this work can be found at `http://opihi.cs.uvic.ca/nime2012gesture`. All the associated software is available upon request. We hope that this will facilitate wider adoption of non-invasive sensing approaches and expand our user base.

[3] `http://cycling74.com/`
[4] `http://www.ableton.com/`
[5] `http://opencv.willowgarage.com/`
[6] `http://www.openframeworks.cc/`
[7] `http://www.ableton.com/maxforlive`

Acknowledgments

Tiago Fernandes Tavares thanks CAPES - Coordenacao de Aperfeicoamento de Pessoal de Nivel Superior - Proc. 2300-11-7 - for funding. The support of the National Science and Research Council of Canada is gratefully acknowledged.

7. REFERENCES

[1] A.Kapur, E.Singer, M. S. Benning, G. Tzanetakis, and Trimpin. Integrating hyperinstruments, musical robots & machine musicianship for north indian classical music. In *Proc. New Interfaces for Musical Expression (NIME)*, New York, NY, USA, 2007.

[2] S. Dahl and A. Friberg. Expressiveness of musician's body movements in performances on marimba. In A. Camurri and G. Volpe, editors, *Gesture-Based Communication in Human-Computer Interaction*, pages 361–362. Springer Berlin / Heidelberg, 2004.

[3] D.Overholt. The overtone violin. In *Proc. New Interfaces for Musical Expression (NIME)*, 2005.

[4] S. Heise and J. Loviscach. A versatile expressive percussion instrument with game technology. In *Proc. IEEE Int. Conf. on Multimedia and Expo (ICME)*, pages 393 –396, 23 2008-april 26 2008.

[5] G. Odowichuk, S. Trail, P. Driessen, W. Nie, and W. Page. Sensor fusion: Towards a fully expressive 3d music control interface. In *IEEE Pacific Rim Conf. (PACRIM)*, 2011.

[6] M. Puckette, T. Apel, and D. Zicarelli. Real-time audio analysis tools for pd and msp. In *International Computer Music Conference*, 1998.

[7] R.Rowe. *Machine Musicianship*. MIT Press, 2004.

[8] T.Machover and J.Chung. Hyperinstruments: Musically intelligent and interactive performance and creativity systems hyperinstruments. In *In Proc. Intl. Computer Music Conference*, 1989.

[9] G. Weinberg and S. Driscoll. Toward robotic musicianship. *Computer Music Journal*, 30(18):28–45, December 2006.

[10] M.J Yoo, J.W Beak, and I. K. Lee. Creating musical expression using kinect. In *Proc. New Interfaces for Musical Expression*, Oslo, Norway, 2011.