A GESTURALLY CONTROLLED IMPROVISATION SYSTEM FOR PIANO

Nicholas Gillian
MIT Media Lab
Cambridge, USA
ngillian@media.mit.edu

Sarah Nicolls
Brunel University
London, UK
sarah.nicolls@brunel.ac.uk

ABSTRACT

This paper presents a gesturally controlled, live-improvisation system, developed for an experimental pianist and used during a performance at the 2011 International Conference on New Interfaces for Musical Expression. We describe the gesture-recognition architecture used to recognize the pianist's real-time gestures, the audio infrastructure developed specifically for this piece and the core lessons learned over the process of developing this performance system.

1. INTRODUCTION

In this paper, we describe the development of a gesturally controlled improvisation system created to facilitate a performer to play the piano while simultaneously controlling custom-designed improvisation software exclusively through gestures. The system was developed by Nicholas Gillian in collaboration with Sarah Nicolls, a UK-based experimental pianist, who performed using an initial version of the system at the 11th International Conference on New Interfaces for Musical Expression (NIME-11)[1]; this performance can be viewed online [1].

As Nicolls has already extensively investigated and written about [2], achieving simultaneous control of audio effects or the parameters of live-processing software whilst continually playing an instrument can be a challenge, even for an accomplished musician. Foot pedals and other forms of dedicated-control interfaces can help achieve this complex synergy, but using these interfaces adds another layer to an already complex physical task. Even if a performer is able to assimilate and interface into their performance environment, it may perturb the musical performance from the audience's perspective, divorcing the intimate connection between performer and instrument. Interfaces can also be costly and/or bulky.

One approach for achieving simultaneous control of digital effects whilst playing an instrument, is to augment the performer with sensors, so as to capture the movements of the player as she plays her instrument, and then use the sensor data to control a real-time performance system (of which there are many examples [3]). A common setup for such real-time augmented-performer based systems is to directly map the sensor data, or derivatives thereof, to control a number of parameters within the real-time audio processing software [4, 5]. While such systems yield a wide range of possibilities for a performer, they can suffer from two significant drawbacks. The first is that the mapping from sensor to parameter is always engaged, that is, *any* movement made by the performer - intended or not - will affect the audio processing software. The second drawback is that these systems often involve the performer manipulating pre-sampled audio, with a pre-planned mapping strategy (though of course this restriction is not a requirement). These drawbacks encumber the application of such systems towards real-time improvisation. Further, such a setup can result in an unnatural, constrained playing style, as the performer tentatively attempts to mitigate triggering a sample from being played at the wrong moment, or the inadvertent adjustment of an audio effect [6]. These aforementioned drawbacks contributed to the motivation for the work presented in this paper, as we now discuss.

[1]http://www.nime2011.org/

2. MOTIVATION

The design of this system was guided by Nicolls' prior extensive experience of using various sensors, such as accelerometers, electromyography (EMG) and pressure sensors, to control complex processing of sampled audio in real time [2]. Several of these projects consisted of either systems that directly mapped the sensor data to a specific effect, or a rudimentary form of gesture recognition, in which a sample was triggered if the value of a sensor exceeded an empirically determined threshold value [2]. Nicolls wanted to use this new system to experiment with the combination of motion-capture technology and more sophisticated gesture-recognition systems, to discover the properties that might be afforded by such technology.

We had four main aims in this project: (1) to free the performer from body-worn sensors or the need to press an external interface; (2) to extend possible control gestures by replacing threshold-based systems with machine-learning based gesture recognition, thus facilitating more complex gestures (i.e. more than a single fixed-point trigger) in multiple locations around the piano playing area; (3) to only use the live piano (including live-sound sampling) as a sound source, to enable an improvisation system that was purely live and intuitive to the performer; (4) to use a relatively cheap, robust, very portable and widely available piece of technology (in comparison with expensive or fragile systems used previously [7]).

3. IMPROVISATION SYSTEM OVERVIEW

The resulting system that we created used a Kinect depth camera to track the pianist's movements, with the data sent via a gesture recognition system to an audio processing system (see Figure 2), allowing the player to create a system of layered and processed loops. This enabled the performer to use hand gestures to 'grab' any theme or motif that had just been played, and 'save' this musical idea to a number of virtual buffers. Numerous setups, processing effects, discrete control gestures and continuous control gestures were experimented with, however just four virtual buffers, one continuous control and one clear gesture were used for the actual NIME-11 performance setup. This configuration was chosen because it yielded a workable compromise, providing an interesting framework within which the performer could improvise without being overwhelmingly complex.

Each virtual buffer was positioned at a predetermined location above the main frame of the instrument (see Figure 1), with each location specified by the performer during a gesture-training phase. Each virtual buffer had a specific audio effect, such as a pitch shifter, granulator, or filter associated with it. After a buffer had been filled with a theme, the performer could return to the theme at any time during the live improvisation, and use a similar gesture to the 'grab' gesture to loop the contained theme. If the theme was already looping, then the same gesture would stop it from looping.

In addition to simply enabling/disabling the playback of a specific theme, the performer could also use a number of continuous hand gestures, such as tilting and rotating their hands, to continuously spatialize a theme, warp and stretch a motif, modify the cutoff frequency of a filter affecting a loop, etc. Finally, the performer could clear the currently recorded themes from all the buffers by making a 'clear' gesture, which consisted of swiping the left hand towards the rear edge of the piano stool. This would fade out any loop that was currently playing, after which all of the recorded themes would be cleared to allow new themes and motifs to be recorded.

The gestural vocabulary for the piece was constructed from a combination of realistic piano-playing inspired gestures, based on the initiate gestures commonly made by pianists to support playing (also referred to as accompanying gestures [8], non-obvious performer gestures [9] and ancillary gestures [10]), and more direct control gestures [11] that have a more obvious action-sound relationship.

Simple discrete controls, such as triggers (play, stop, record), were designed to exist within the physical area that would be normally be accessed whilst playing and we aimed to operate these with innate-looking gestures (such as hands raising above the keyboard, as may happen after playing a loud chord or staccato gesture). More advanced continuous controls (i.e., pitch-bending, timewarping) would occur with gestures placed similarly within

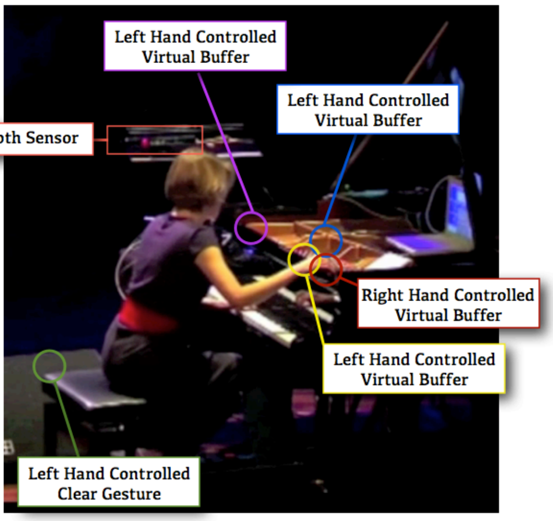

Left Hand Controlled Virtual Buffer
Left Hand Controlled Virtual Buffer
Depth Sensor
Right Hand Controlled Virtual Buffer
Left Hand Controlled Virtual Buffer
Left Hand Controlled Clear Gesture

Figure 1. Sarah Nicolls using the gesturally controlled improvisation system at a performance during NIME 2011. The image also illustrates the location of the four virtual buffers and the location of the 'clear' gesture.

the playing area, but now with more direct and obvious action-sound relationships, such as rotating the hand in mid-air to control the pitch-bending of a loop, placing them outside of a pianist's normal performing gestural vocabulary.

The primary performative aim behind this piece was to achieve a relationship between the performer's hands and the resulting sound that was motivated most strongly by how it made sense to the performer from a control point of view, such as rotating the hand down to the left to slow the speed of a loop once it was playing, rather than from the audiences perspective. As we set out to use a combination of the gestures listed above, however, we anticipated creating a system that was controlled by a fairly obvious action-sound mapping, thus making it more expressive both for the player and the audience (as suggested by Fels et. al. [12]).

4. GESTURE RECOGNITION

The technical aspect of the gesture-recognition component of this live improvisation system can be segmented into three distinct elements; namely tracking, gesture classification, and mapping.

4.1. Tracking

The performer's movements were tracked using a Kinect depth sensor[2] and an open-source skeleton-tracking library[3]. This tracking infrastructure significantly reduced the complexity of estimating the three-dimensional position of the performers body joints (compared with, for instance, a more conventional computer-vision based tracking sys-

[2]Microsoft Kinect SDK: `http://www.microsoft.com/en-us/kinectforwindows/`
[3]OpenNI: `http://openni.org/`

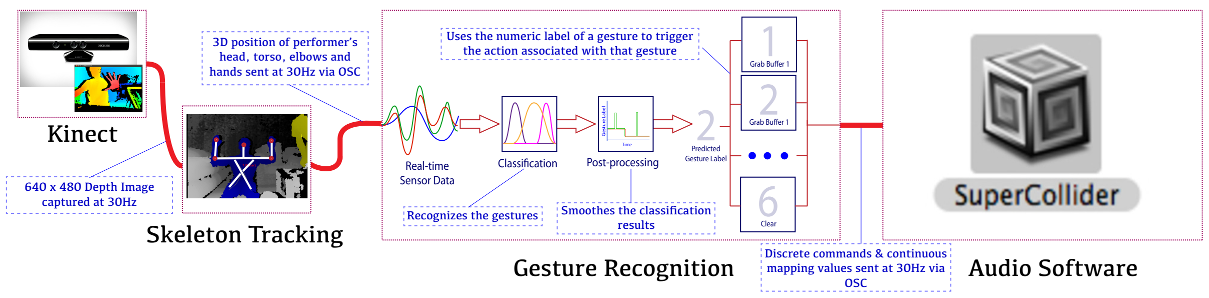

Kinect
640 x 480 Depth Image captured at 30Hz
3D position of performer's head, torso, elbows and hands sent at 30Hz via OSC
Skeleton Tracking
Uses the numeric label of a gesture to trigger the action associated with that gesture
Real-time Sensor Data
Classification
Post-processing
Predicted Gesture Label
Recognizes the gestures
Smoothes the classification results
Grab Buffer 1
Clear
Gesture Recognition
Discrete commands & continuous mapping values sent at 30Hz via OSC
SuperCollider
Audio Software

Figure 2. The infrastructure created for this gesturally controlled improvisation system, showing the tracking, gesture recognition and audio software.

tem, such as that used in [13]). However, specific performance requirements did pose some interesting tracking challenges that needed to be overcome.

These challenges were mainly a result of the Kinect depth sensor being applied in a somewhat unconventional tracking environment. The Kinect is primarily designed for tracking a user's movements while they are standing in an uncluttered environment, such as a living room, facing the device. Tracking a performer while they are seated at a piano could, therefore, result in a number of unwanted sporadic joint-estimation errors during a performance. These errors are commonly a result of the performer's hands being occluded by another body joint, or part of the piano body itself.

The number of joint-estimation errors were significantly reduced by experimenting with the placement of the depth sensor, in relation to the performer and piano. In this instance, the optimal position of the depth sensor was found to be approximately orthogonal to the pianist (see Figure 1), with the sensor elevated high enough to provide an unobstructed view of both the performer's hands over the keyboard region, and the performer's extended hand gestures above the pianos' tuning pins (where the music stand would normally be). Any sporadic joint-estimation errors not resolved by careful positioning of the depth sensor were removed by applying a jerk filter and low-pass filter to the raw inferred joint positions. The tracking software then sent the filtered three dimensional inferred joint positions of the performer's head, torso, elbows and hands at 30Hz, via Open Sound Control (OSC) [14], to the gesture recognition software for classification. It should be noted that a number of new tracking libraries, such as the Microsoft Kinect SDK, have been released since the initial debut performance of this system that further mitigate these tracking problems.

4.2. Gesture Classification

The gesture-recognition component of this live-improvisation system was created using the SARC EyesWeb Catalog (SEC) [15], an extension to the free graphical-development environment EyesWeb [16]. The SEC features a large number of machine-learning and signal-processing algorithms, all of which have been designed for real-time gesture recognition. Similar to other graphical-development environments, such as MAX[4] or Pure Data[5], algorithms are represented in EyesWeb by blocks, which can be connected together on a patch window to form a signal chain. The SEC gesture-recognition algorithms are also encapsulated as blocks, which therefore enables a user to easily create their own gesture-recognition system by connecting a small number of blocks together, as illustrated in Figure 3.

Unlike the previously mentioned empirically set threshold-based recognition systems, the user does not explicitly hand code a machine to recognize a gesture in machine-learning based gesture-recognition systems. Instead, a machine-learning algorithm is used to automatically infer the relationship between a performer's movements, captured by a sensor, and a gesture. A machine-learning algorithm performs this inference during a learning (or *training*) phase, by directly analyzing the sensor data recorded as the user performers a number of example gestures; resulting in a trained classification *model*. This model can then be used to classify the category (i.e. gesture) of new, previously unseen, data. This classification can occur in real-time and can therefore be used to recognize if a player is currently performing a gesture. The process of creating a gesture-recognition system can therefore be broken down into three steps: (1) recording some examples of each of the gestures that need to be recognized; (2) training a classification model using a machine-learning algorithm and the recorded training data; (3) using the classification model to recognize a performer's real-time gestures. The SEC has a number of blocks to support all three of these steps.

The exact form of the machine-learning algorithm used for training and classification depends on the context of the recognition problem the user is attempting to solve. For instance, the type of algorithm used to recognize a static posture, such as if a performer has their right hand in the air, will commonly be different from that used to recognize a temporal gesture, such as a performer waving their hand. An Adaptive Naïve Bayes Classifier (ANBC) [17] algorithm was used to recognize the grab, save, and clear gestures for this live-improvisation system, as these gestures are essentially static postures - i.e. either the performer has her hand in the grab location, or she does not. The ANBC algorithm is particularly applicable for rec-

[4]MAX: `http://cycling74.com/products/max/`
[5]Pure Data: `http://puredata.info/`

ognizing such gestures because it can be trained rapidly (i.e. ∼1 second for training the model used for this performance) with just a few seconds of user-supplied training data for each gesture.

The ANBC algorithm essentially works by fitting K Gaussian distributions to the sensor data, or features computed from this data, for each of the K gestures (i.e. grab buffer 1, grab buffer 2, clear, etc.), recorded by the performer. The ANBC algorithm is a supervised learning algorithm, which means that the user needs to hand label the sensor data with a numeric value representing the corresponding gesture the data is associated with. Training data can easily be labelled using the SEC, as the user simply needs to set a numeric parameter value of the labelled training tool block prior to recording the training data for gesture k. A gesture can then be classified in real time by projecting the new sensor data through each of the K Gaussian distributions and computing which distribution gives the maximum likelihood (i.e. highest value). The ANBC predict block will then output the numeric label associated with the predicted gesture, along with the probabilities of each of the k gestures given the current input-sensor data. The input to the ANBC algorithm consisted of a vector containing the three dimensional inferred joint positions of the performers head, torso, elbows and hands.

Of course, a player is not always performing a gesture. Fortunately the ANBC algorithm can automatically reject non-gestural movements, such as normal piano playing, by comparing the current maximum likelihood value with the average maximum likelihood of the other k gestures in the training data. If the current maximum likelihood is within a given distance of that of the gestures in the training data then the gesture with the maximum likelihood will be classified as gesture k, otherwise it will be rejected. The automatic rejection capabilities of the ANBC algorithm are of great benefit to a performer as it enables her to record a few seconds of training data for each gesture she wishes the system to recognize, without having to create an additional 'null-gesture' and record a large amount of data to ensure the system knows what is *not* a valid gesture.

To further increase the robustness of the real-time gesture classification, the predicted gesture label of the ANBC algorithm was first post-processed using a SEC Class Filter block. The class filter will only output a gesture label when n consecutive gesture labels are observed within a predetermined time window (i.e. 100ms), outputting the null-gesture label of 0 otherwise. This post-processing step helped mitigate false-positive recognition errors and stopped erroneous triggers if, for instance, the performer momentarily passed her hand through one of the gesture areas whilst moving to another area of the piano. If a valid gesture label was output by the Class Filter block, the numeric value of the label would trigger the corresponding gesture to be actioned. The action would then be sent to the audio processing software via OSC to trigger the appropriate audio event, such as grab and save the current theme in virtual buffer 1.

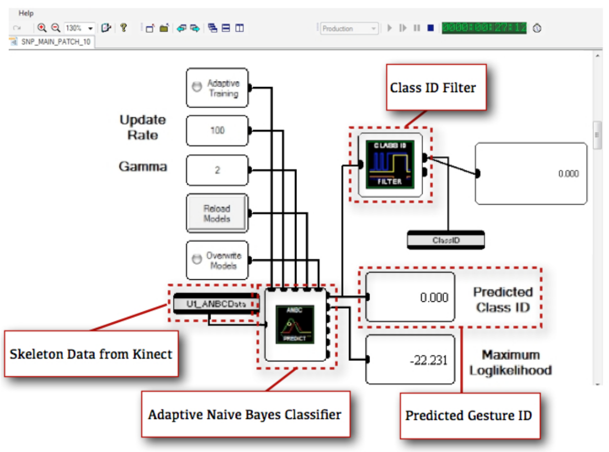


Figure 3. A small section of the gesture recognition patch in EyesWeb. The blocks visible show the Adaptive Naive Naïve Classifier algorithm that was used to recognize the grab, play, and clear gestures.

4.3. Mapping

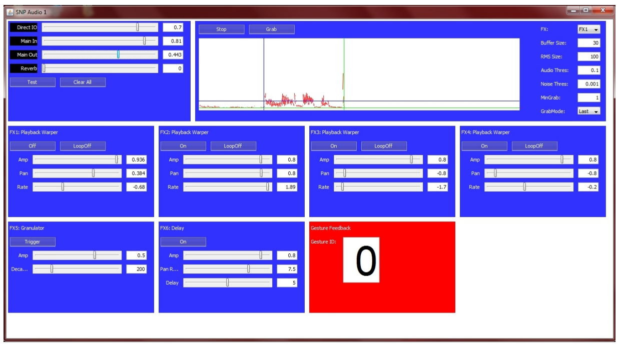
In addition to recognizing the performer's discrete control gestures (grab, loop, clear, etc.), the SEC was also used to continually map the performer's movements to a number of specific effects. This continuous mapping was combined with the discrete gesture recognition, which meant that a performer's movements were only continuously mapped to an effect (such as the cutoff value of a filter) when the performer had their hand in a specific 'control area'. The mapped continuous effect value would then be sent to the audio-processing software via OSC.

Combining the continuous mapping with the discrete gesture recognition therefore resolved the problematic issue of the mapping from a sensor to an audio parameter always being engaged, as the continuous mapping was only enabled if the performer first made the correct discrete gesture within the control area. The continuous mapping value would then be locked at its current value if the performer moved away from the control area.

5. AUDIO INFRASTRUCTURE

The audio infrastructure for this live-improvisation system was developed using SuperCollider [18], an environment and programming language for real-time audio synthesis and algorithmic composition (see figure 4). The input to the audio system consisted of the live audio from the piano captured by a stereo close-mic setup. This stereo signal was fed into a circular buffer, which continually recorded the last thirty seconds of live audio. If the performer made the 'grab' gesture in a free virtual buffer, a signal would be sent from EyesWeb to SuperCollider via OSC indicating that the last distinct theme or motif should be cut and saved to the respective buffer using a naïve segmentation algorithm.

A specific theme or motif was segmented from within the circular buffer by first computing the root mean squared



Direct IO 0.7
Main In 0.81
Main Out 0.443
Reverb 0

FX: FX1
Buffer Size: 30
RMS Size: 100
Audio Thres: 0.1
Noise Thres: 0.001
MinGrab: 1
GrabMode: Last

FX1: Playback Warper — Off, LoopOff; Amp 0.936; Pan 0.384; Rate -0.68
FX2: Playback Warper — On, LoopOff; Amp 0.8; Pan 0.8; Rate 1.89
FX3: Playback Warper — On, LoopOff; Amp 0.8; Pan -0.8; Rate -1.7
FX4: Playback Warper — On, LoopOff; Amp 0.8; Pan -0.8; Rate -0.2
FX5: Granulator — Amp 0.5; Deca... 200
FX6: Delay — On; Amp 0.8; Pan R... 7.5; Delay 5
Gesture ID: 0

Figure 4. The real-time audio improvisation interface created using SuperCollider.

(RMS) value of the buffer's normalized audio signal. This RMS signal was then scanned in reverse order to detect the end of a theme, given by the RMS signal exceeding an empirically-set threshold. The remainder of the RMS signal was then scanned to locate the start of a theme, given by the RMS signal dropping below a second empirically-set threshold. If both these events were detected then the audio data between the two events would be cut and stored in the chosen virtual buffer. If either of these events were not detected then the segmentation algorithm would simply cut the audio from one event until the other end of the audio buffer, adding a small fade at the start of the audio to mitigate any audio clicks.

This automatic segmentation algorithm freed the player from first having to hit a button or perform a specific 'start of theme' gesture, before she improvised a new theme. It additionally liberated the performer from having to make a new theme fit within a predetermined loop length (the 30 second circular buffer length was set as the performer imagined this would cover at least twice the length of any possible improvised theme, thus ensuring that any new theme could be adequately segmented from the buffer).

6. DISCUSSION

Our methodology for developing this system entailed several laboratory sessions and we began our work with a Polhemus[6] six-degree of freedom magnetic tracking system. We knew from the outset this would need to be replaced by technology that met our criteria of being portable, relatively cheap and easily available and we soon replaced the Polhemus with a Kinect. Building the set up from scratch, we made all of the decisions referred to in this paper as we went along but for the performance itself we were forced by time constraints to shrink back some of our more ambitious ideas that had been previously explored during the lab sessions. This meant most crucially that in the end we were only able to include one complex gesture in the system Nicolls performed with at NIME and also that we had to change the placement of the discrete gesture trigger locations (e.g. the grab buffer 1 gesture). The resulting performance had a lot more normal unadulterated playing (i.e. without loops layering etc) than we might have aimed for with more time.

[6]http://www.polhemus.com/?page=Motion_LIBERTY

6.1. Gestural Interaction

The enabling of innate control, using realistic, ancillary, gestures (i.e. ones that are already part of the pianists movements, such as moving a hand upwards from the playing position to move to a new area of the keyboard) allowed for a system where no extra physical language had to be learned (unlike with e.g. EMG sensors) and where the pianist could move effortlessly between instrumental and ancillary (control) gestures without having to make explicit changes in her physical state. This mirrored the natural control language used in Jonathan Green's *Into Movement* [6], with the crucial difference that this time, as control points were more defined in a 3-dimensional space, the performer could easily turn off the sensors by not performing the associated gesture in the pre-specified control areas. However, for the NIME-11 performance we ended up having to create some of the trigger points inside the piano, over the tuning pins, so that they would be more secure to find. Ironically this lent a more theatrical element to simple triggers and thus perhaps undermined the system's spectrum of subtle, ancillary, gestures to those more flamboyant or demonstrative performative ones.

This combination of subtle gestures with larger, performative ones led to confusion. An already problematic result of a system relying on innate ancillary gestures is the difficulty the audience may have in detecting the control function, leading to frustration, especially perhaps in an academic setting such as NIME, where the control functions are the lifeblood of the research taking place. Indeed, the lack of detection was noted by several audience members and relayed to us through informal feedback. However, combined with the highly performative continuous gesture (seen at 3'14 and 3'20 in the online video [1] in Nicoll's left arm), the confusion was doubled. This continuous gesture had to be performed more elaborately than we would have liked, to ensure reliable reading by the Kinect. Even so, at serval points during the performance (for example at 8'21 in the video) the continuous gesture control was not enabled correctly, a result of the discrete gesture recognition that engaged the continuous control not being detected. This resulted in Nicoll's performing an increasingly extravagant movement that was not linked to any obvious change in the audio, creating a disconnect.

6.2. Comparison With Other Gestural Systems

Comparing the system we created to those Nicolls had previously commissioned [2], we had several observations. There could potentially be a wider gestural vocabulary, as our system could easily imitate both body-worn and fixed-point systems, through manipulation by the performer. However, the urgent, emotional quality of the EMG sensors was lacking, so that actually the imaginative creation of imitative gestures would be perhaps rather hollow here.

Of course, EMG could easily be integrated into the current system to compliment the current sensor infrastructure, however, this would violate the first main aim of this project (see section 2).

The software we were using could also allow for the development of more complex gestures but the control of the live audio processing needed to be much more developed to make use of the software to its full extent. Muscularly speaking, the performer was much freer than with EMG, with no need to create tension to trigger events. There was still a gestural language that needed to be learnt, however and although this didn't demand physical tension from the performer, it did require leaping to fixed points in the performance space while also maintaining an improvisational flow. Competent control of the layering of the loops needed more practice so that the control functions could be absorbed into the playing style in a convincing way, for example, dealing with the challenge of controlling a buffer placed over the top register while playing in the lowest register of the piano.

Finally, there was a much bigger potential readable space than with fixed haptic sensor systems - the whole of the reachable space from the piano stool, including mid-air, was available to use. On the other hand, predetermining static points in the air, as opposed to the control enabled by body-worn sensors (which can be used anywhere in the space), created a different sort of rigidity and still tethered the performer in a similar way as if buttons had needed to be pressed.

6.3. Only Using Live Piano Sound

For the performer, this system represented a return to the sonic palettes used by Richard Barrett in *Adrift* (2007) [19] and Luigi Nonos *...sofferte onde serene...* (1976) [20] in the close matching of piano sounds to the electronic part. Using only live piano sound enabled a totally instinctive improvisation system whereby a self-referential texture could easily be built up using genuinely improvisational methods (i.e. not tethered to previously composed pre-recorded samples). The intuitive nature of this for the performer enabled each piece generated by the system to be unique in its sonic language, meaning that the scope for the system was in theory much greater than others previously used by the performer, potentially creating live and from scratch a whole concert of pieces using the same technological set up. In further sonic comparison with pre-recorded sample systems, it was fundamentally easier to create a cohesive sound world, with all sounds stemming from the same source. There was a more intuitive sense of where the performer was in relation to the controlled sounds, having just created them herself live, and this in turn perhaps made the live creation of the piece easier to comprehend for the audience. Musically, the looping system allowed a large variety of sonic results from a simple base.

6.4. Visual Feedback

One of the key lessons that was learned during the iterative development of this system was the crucial importance of some form of visual feedback, to compensate for the lack of haptic information in triggering the control areas. Visual feedback, indicating to the performer the current state of the system, such as if a control gesture was recognized or which virtual buffer was filled or playing, needed to be clear and easy to read quickly in performance. The feedback itself very quickly developed in the laboratory sessions as we added color and shape features to enable very quick, peripheral understanding of the information. Because one screen of the computers running the real-time performance system was already used to check the calibration and receive data from the Kinect, this actually led to both laptops being used inside the piano, leading to a rather bloated technological demand in terms of set up.

6.5. General Technical Difficulties or Limitations

The initial debut performance of this live-improvisation system at NIME-11 featured a number of general technical difficulties and limitations that could be improved in future versions of the system. For instance, as previously mentioned, two computers were required to run the real-time performance system, with the first computer running the tracking and gesture-recognition software, and the second computer running the real-time audio software and visual feedback in SuperCollider. While this two-computer infrastructure greatly reduced the computational overhead on both machines, which significantly abated the chances of either the tracking, gesture, or audio software momentarily freezing or stuttering during the live performance, two computers did create an additional technical burden on the piece.

There were also two staging issues, both related to using the Kinect depth camera. Firstly, the software used to track the performer's movements required a calibration phase, during which the performer had to make a specific calibration pose and hold this for a number of seconds. This calibration step had to be repeated each time the performer left the Kinects field of view, resulting in Nicolls having to remain on stage prior to performing the piece. This in turn gave a theatrical element which we had not sought and which in some cases confused audience members, seeming a deliberate choice. Secondly, the Kinect demands a specific set up (see section 4.1) and this requires a minimum of space, with extra stage furniture also required to balance the Kinect on. The placement of the Kinect was further complicated by the use of extended playing techniques, which meant that the depth camera had not only to be positioned so as to view the performer as she played the keys, but also had to be capable of viewing the performers extended hand gestures inside the body of the piano itself.

As previously noted, a number of new tracking libraries have been released since the initial performance at NIME-11, which significantly help to mitigate the calibration issues. Nonetheless, setting up the tracking of a performers movements could be further improved in future versions of the system.

7. FUTURE WORK

The authors would like to pursue refining this system because it has much potential as an easily transferable improvisation system to generate many different pieces. One of the key features that could be explored more fully is the fact that the user does not need to explicitly hand code the machine to recognize a gesture. Being able, as a non-programmer, to teach the computer complex gestures is a huge advantage and could be exploited much more. The benefit of having only one fixed piece of sturdy, portable and easily replaceable technology cannot be underestimated, and the potential advantage of the performer to be able to re-map the system unaided is highly advantageous. In particular, the possibility to transfer the system to Nicolls Inside-Out Piano is of great interest, although positioning the Kinect at the top of the instrument looking down (thus creating a very compact system, easily portable without repositioning the Kinect in relation to the instrument) has had to be disregarded at this time due to the calibration requirements.

8. CONCLUSION

This was found to be a very good potential system allowing the creation of any new piece by the performer without having to re-program at all or create pre-recorded samples. It would be easy to calibrate in new ways and so be easily adaptable to new theatrical, gestural or musical ideas. However, it would be most effective in a simplified technical state where only one computer was required and where calibration could somehow be stored to allow freedom on stage in a concert situation. It is our conclusion that this system could be used effectively to create pieces built from a wide range of gestural vocabularies. From gestures with an obvious action-sound mapping, to more initiate, ancillary, gestures. This could be played with to discover the full poetic potential within the system and to test how these gestures influence both the performer's and audience's perceptions of the piece.

9. REFERENCES

[1] S. Nicolls, "Live improvisation performance viewable online at https://vimeo.com/26678719. nime2011, oslo, norway," May 2011.

[2] S. Nicolls, *Interacting with the piano*. PhD thesis, Brunel University School of Arts, 2010.

[3] E. Miranda and M. Wanderley, *New digital musical instruments: control and interaction beyond the keyboard*, vol. 21. AR Editions, Inc., 2006.

[4] Q. Yang and G. Essl, "Augmented piano performance using a depth camera," in *Proceedings of NIME*, 2012.

[5] W. Brent, "The gesturally extended piano," in *Proceedings of NIME*, 2012.

[6] J. Green, "Into movement. broadcast on hear and now, bbc radio 3," August 2007.

[7] S. Nicolls, "Twenty-first century piano," in *Proceedings of NIME*, 2009.

[8] F. Delalande, "La gestique de gould," *Glenn Gould Pluriel*, pp. 85–111, 1988.

[9] M. M. Wanderley, "Non-obvious performer gestures in instrumental music," *Gesture-based communication in human-computer interaction*, pp. 37–48, 1999.

[10] M. Wanderley and P. Depalle, "Gestural control of sound synthesis," *Proceedings of the IEEE*, vol. 92, no. 4, pp. 632–644, 2004.

[11] D. McNeill, *Language and gesture*. Cambridge University Press Cambridge, UK, 2000.

[12] S. Fels, A. Gadd, and A. Mulder, "Mapping transparency through metaphor: towards more expressive musical instruments," *Organised Sound*, 2002.

[13] F. Huang, Y. Zhou, Y. Yu, Z. Wang, and S. Du, "Piano ar: A markerless augmented reality based piano teaching system," in *Proceedings of the 2011 International Conference on Intelligent Human-Machine Systems and Cybernetics (IHMSC)*, 2011.

[14] M. Wright, "Open sound control: an enabling technology for musical networking," *Org. Sound*, vol. 10, pp. 193–200, Dec. 2005.

[15] N. Gillian, R. B. Knapp, and S. O'Modhrain, "A machine learning toolbox for musician computer interaction," in *Proceedings of NIME*, 2011.

[16] A. Camurri, P. Coletta, G. Varni, and S. Ghisio, "Developing multimodal interactive systems with eyesweb xmi," in *Proceedings of NIME*, 2007.

[17] N. Gillian, R. B. Knapp, and S. O'Modhrain, "An adaptive classification algorithm for semiotic musical gestures," in *Proceedings of the 8th Sound and Music Computing Conference*, 2011.

[18] J. McCartney, "Rethinking the computer music language: Supercollider," *Computer Music Journal*, vol. 26, no. 4, pp. 61–68, 2002.

[19] R. Barrett, "Adrift. [psi 09.10] london. commercially available at http://www.emanemdisc.com/psi09.html," August 2009.

[20] L. Nono, "sofferte onde serene' ricordi #r132564, italy," 1976.