Multi Sensor Tracking for Live Sound Transformation

Anton L. Fuhrmann
VRVis Forschungs GesmbH
Donau-City-Strasse 1
Vienna, Austria
fuhrmann@vrvis.at

Johannes Kretz
Center for Innovative Music Technology ZiMT
University for Music and Performing Art Vienna
Anton-von-Webern-Platz 1
Vienna, Austria
zimt@mdw.ac.at

Peter Burwik
XXth Century Ensemble
Vienna
Hügelgasse 12
Vienna, Austria
exxj@aon.at

ABSTRACT

This paper demonstrates how to use multiple Kinect™ sensors to map a performer's motion to music. Skeleton data streams from multiple sensors are merged in order to compensate for occlusions of the performer. The skeleton joint positions drive the performance via open sound control data. We discuss how to register the different sensors to each other and how to smoothly merge the resulting data streams and how to map position data in a general framework to the live electronics applied to a chamber music ensemble.

Keywords

kinect, multi sensor, sensor fusion, open sound control, motion tracking, parameter mapping, live electronics

1. INTRODUCTION

In 2006 Peter Burwik and the *XXth Century Ensemble Vienna* (exxj)[1] started a project with the goal of developing an interactive environment for a dancer and an instrumental ensemble, interacting through live electronics. The main idea was to track the dancer's movement and to use the motion data for the control of live electronic sound transformation of the ensembles instrumental playing. At early experimentation stage Peter Burwik invited Johannes Kretz to participate and consequently to develop the technical basis for further improvement. Since Kretz became director of the *Center for Innovative Music Technology* (ZiMT)[2] of the University for Music and Performing Art Vienna, "exxj" and ZIMT cooperated in several concerts.

2. OBJECTIVES

The particularities of the project require a tracking system with the following characteristics:

High Precision A relatively high precision of motion tracking is needed, because even small movements of the dancer should have noticeable effects on the acoustic results. This is due to the fact that when chamber musicians play contemporary music, the sensitivity and accuracy of the musician's actions and their sounding results are usually extremely subtle and accurate. Therefore a system which would require relatively large movements of the dancer (in the range of 20 cm or more), would create a disturbing imbalance between the expression of the dancer and the ensemble and therefore not deliver satisfying artistic results.

Low Latency For the same reason, the latency of the system should be relatively small, so that the causality of the acoustical effects introduced by the dancer's movements are obvious to the audience.

Ease of Use The system should ideally be easy to set up and to transport (for tours of the ensemble). The performer should be able to move freely, without cables or unwieldy tracking targets.

Low Cost As always, total costs are a determining factor. Most commercially available motion capturing systems cost $10.000 and more. We wanted an inexpensive system consisting of easily replaceable, off-the-shelf components.

3. PROPOSED SOLUTION

After experimenting with Local Position Measurement through radio waves [12] and optical infrared tracking with rigid-body targets [9], we finally decided to explore a system of connecting multiple Kinect sensors, which is less expensive than the above mentioned systems (by magnitudes) and still offers a reasonable quality and stability of tracking data.

The inexpensive Kinect sensor [7] has already been proposed as a musical interface device [13], and it does not require any rigid-body targets, which improves the visual aesthetics of the performance a lot.

For our project we use skeletal information from the sensor, especially the position of the skeletons extremities: head, hands, and feet. The Microsoft Kinect™ SDK provides real-time segmentation of the sensors depth images into one or maximally two skeletons, but cannot overcome the sensors inherent shortcomings: if not all joints of the tracked person are visible, missing joint information is at best inferred by the software (in most cases an approximation based on visible surrounding joints), but in many cases just marked as "not tracked". While these drop outs are an inherent property of all optical trackers, most non depth camera based trackers reduce the occurrence of occlusions by using multiple cameras in the first place. By using more than one sensor, we can reduce occlusion and improve the stability of our tracking set-up (Figure 1).

[1] http://www.exxj.net/
[2] http://www.mdw.ac.at/zimt/

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.
NIME'13, May 27 – 30, 2013, KAIST, Daejeon, Korea.
Copyright remains with the author(s).

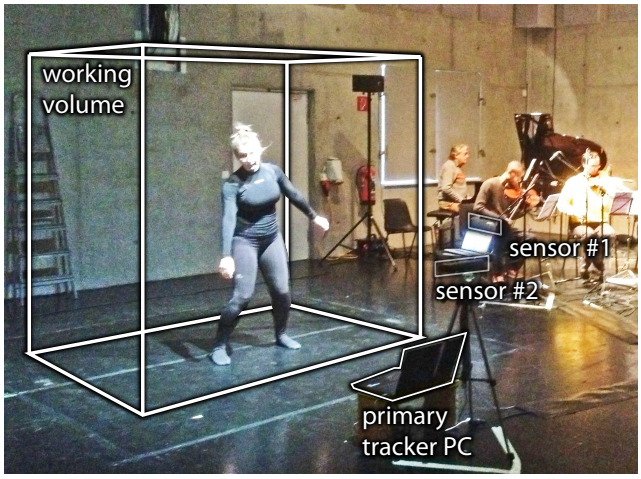

Figure 1: Physical set-up of the performance. Two sensors at 90°to each other extend the working volume of the trackers.

Previous work [1, 6] has shown that employing multiple Kinect sensors with overlapping tracking volumes works, albeit at reduced precision. By mechanically shaking the sensors relative to each other, the interference between the sensors can be reduced, but this method produces too much acoustic noise for use in musical performances. These interference rises with the number of sensors, respectively their projected infra-red patterns, but can be minimized by keeping the sensors axis at a minimum angle to each other. For our purposes, we have determined that an angle of about 90°works in satisfactory way, both with respect to sensor interference and to occlusion reduction.

4. MERGING SENSOR STREAMS

Since we now have two or more data streams from non-collocated sensor origins, we have to merge these redundant streams into one stable skeletal data stream. For this, there are two steps in the processing queue, where we could merge:

4.1 Merging point data

The Kinect sensors deliver depth images at a resolution of 640×480 pixels at 30Hz. These images can be interpreted as point clouds in 3D space. When the position and orientation of multiple sensors is known, their point clouds can be merged to a single cloud. This resulting cloud can then be treated to similar segmenting and skeletonizing operations as the ones normally applied to the source depth images. Such a method is used by iPi Soft [5] in their motion capturing product. Their results are impressive, but their implementation works only in an offline post-processing step, while we need a real-time response with minimal delay.

Wilson et al. [14] have used merged point clouds for interaction, but are using only the minimally necessary information for collision detection on planar surfaces, thereby avoiding the generation of a skeleton completely.

Since we did not have neither time nor resources to implement a new real-time skeletonizing algorithm for merged point clouds, we opted for merging the already segmented skeleton data streams.

4.2 Merging skeleton data

Since Microsoft's Kinect™ SDK [7] delivers up to two full skeleton data streams per sensor in real-time, merging the already skeletonized data seemed to us to be a sensible and - given the time constraints - quickly implementable approach. While Kinect depth images contain about 40.000 pixels per person, the data of a complete skeleton contains only the 3D positions of 20 joints, a more manageable amount.

Since the skeleton data stream from the Microsoft SDK is already filtered by *Holt Double Exponential Smoothing* [2], which delivers satisfactory results, we refrained from implementing our own post-filter for the position data.

Naive merging, which we implemented first, just takes the mean of all available data points the sensors deliver for a specific joint. While this approach works fine most of the time, it also leads to jumps in the output values every time a sensor loses a joint or resumes tracking it. These jumps result from systematic errors depending on body and limb orientation relative to the different sensors (visible in Figure 3).

To compensate these systematic differences, we used a simple empirical approach: we smoothed the difference vector resulting from a vanishing or re-appearing skeleton joint over a small time interval (<300ms), thereby converting sudden jumps in output values to a short period of drift (Figure 2).

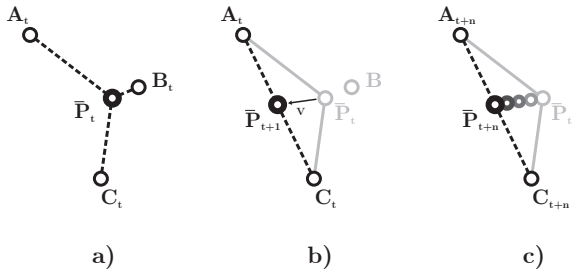

Figure 2: Interpolated merge of points.

Figure 2a shows the merged point $\bar{P}_t$ at time t as geometric mean of A_t, B_t, and C_t, the three joint positions delivered from the three sensors at time t.

Figure 2b shows the mean after B vanishes, e.g. by sensor B being occluded. The merged point $\bar{P}_{t+1}$ jumps to the new position along vector v.

Figure 2c shows the result of linearly interpolating the results position along vector v from time t to $t+n$: the same result position as in figure 2a is reached after n intermediary steps, avoiding a sudden jump in the resulting merged position.

Points P_1 and P_3 normally change position during the interpolation interval $[t, t+n]$, so that the final position $\bar{P}_{t+n}$ in b) is of course not the same as $\bar{P}_{t+n}$ in c).

In detail, this means that the position vector $\vec{p}$ of P is calculated from the joint position vectors $\vec{a}$, $\vec{b}$, and $\vec{c}$:

$$\vec{p}_t = \frac{\vec{a}_t + \vec{b}_t + \vec{c}_t}{3}$$

and at time $t+1$, when sensor B has dropped out:

$$\vec{p}_{t+1} = \frac{\vec{a}_t + \vec{c}_t}{2}$$

which leads to a difference vector of:

$$\vec{v} = \vec{p}_{t+1} - \vec{p}_t$$

and the interpolation from t to $t+n$ is calculated as:

$$\vec{p}_{t+i} = \frac{\vec{a}_{t+i} + \vec{c}_{t+i}}{2} + \left(\frac{n-i}{n}\right)\vec{v}$$

A higher order interpolation is possible, but would only lead to complications when another change in sensor data availability should happen during the interpolation interval. If

this happens now, the calculation simply restarts, using the last valid state of the interpolation output instead of $\vec{p}_t$ to calculate $\vec{v}$.

5. SENSOR REGISTRATION

For the sensor fusion to work, we have to transform all sensor skeleton data into the same coordinate system. This *registration* process has to be performed every time the sensors' position is changed, e.g. every time a new performance is set up. Our proposed method takes only a few minutes and can be easily performed through the user interface. Input to the registration algorithm are synchronized point clouds: over a given time interval, we collect all available skeleton data. Corresponding joint positions collected at the same time from different sensors give us one registration data record per joint per time sample. For our set-up we use the coordinate system of one selected (primary) sensor and register all other sensors' coordinate systems to it. This reduces the problem to a registration between one or more pairs of coordinate systems, which is the well known *absolute orientation problem*, which can easily be solved using Horn's method [3].

The execution of the registration procedure is simple: while the data is collected, the performer moves slowly through the overlapping field of view of the sensors, moving his or her arms slowly to cover a large volume of data points. The data collection procedure terminates, as soon as enough point correspondences (about 200) have been collected. The time needed for this varies slightly because of temporary occlusions, but normally it takes about one minute, since we are only using every tenth sample from the sensors to get well-spaced points.

The registration transform itself is calculated in a few seconds and the data is saved in a set-up specific file, which can be reused until the positions of the sensors are changed.

The quality of the result is returned as a mean error value (usually about 2-3 cm) and an optional interactive error visualization (Figure 3). There are two sources for these position errors: the non-linearity of the sensor's measurements as documented in previous work [13], and the asynchronous operation of the sensors, which can lead to sample errors of up to $1/30^{\text{th}}$ of a second.

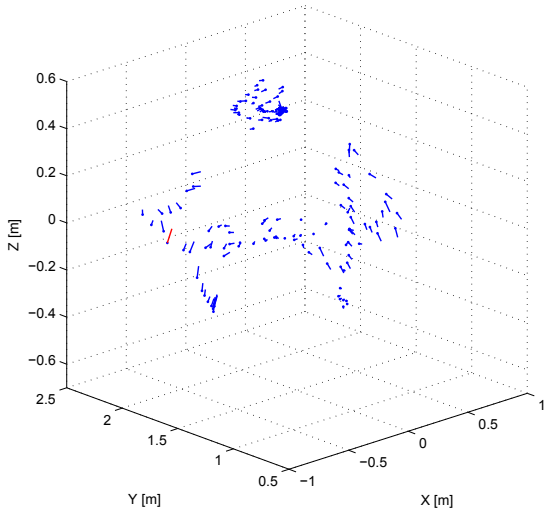

Figure 3: Visualization of the registration results. The top point cloud is the users head, below are the overlapping clouds of the hands.

6. IMPLEMENTATION

The Microsoft Kinect™ SDK provides a managed API, which made an implementation in C# the preferred choice. Only for the implementation of Horn's method additional Matlab code was used.

We opted for a distributed approach using one PC per sensor for two reasons: first, the high throughput of the USB connection does not lend itself to extension cords, making a single, central PC in a physically larger set-up impractical; and second, the SDK only supports one sensor as an input device, multiple sensors can only be accessed alternately.

The resulting set-up is illustrated in the block diagram in Figure 4: the sensors are connected to their respective PCs via relatively short USB 2.0 cables and the PCs are connected to each other via LAN. We normally use ethernet cabling, but WiFI is of course possible, if the environment allows.

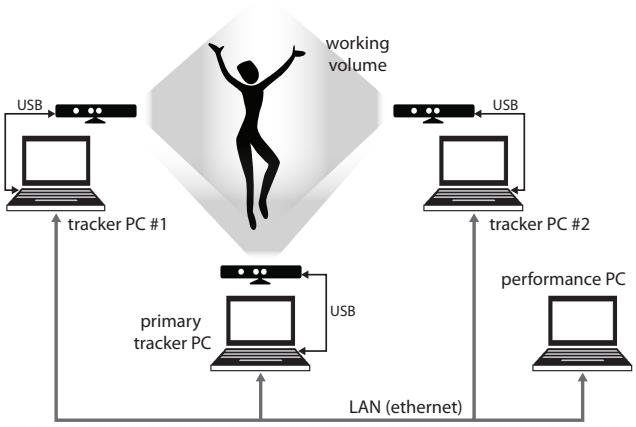

Figure 4: Block diagram of the set-up.

All communication via LAN uses the *Open Sound Control* (OSC) protocol [11]. Currently, we mostly use the joint information of the extremities - head, hands, and feet - since interior joints of the skeleton have proven to be tracked not precisely enough. We use the same format for messages between the tracker PCs and between the primary tracker (which implements the data fusion) and the performance driving PC. This enables a simple set-up with only one tracker to be used as test installation, eliminating the registration and fusion steps completely.

7. APPLICATION FOR THE CONTROL OF LIVE ELECTRONIC MUSIC

The skeleton data sent from the merging software are position data of numerous joints in 3 dimensions: hip center, spine, shoulder center, head, left shoulder, left elbow, left wrist, left hand, right shoulder, right elbow, right wrist, right hand, left hip, left knee, left ankle, left foot, right hip, right knee, right ankle, right foot. These data are transmitted into the Max/MSP environment [15] via Open Sound Control (OSC [7]) using UDP[3].

A general framework in Max/MSP was developed at ZiMT by Johannes Kretz in close collaboration and intensive testing with The XX. Century Ensemble Vienna for applying the sensor data on various musical parameters of the live electronic transformations of the instrumental sounds. While the framework had to be designed in a way which allows different composers with different aesthetics to imple-

[3]http://tools.ietf.org/html/rfc768

ment their work, certain general constraints of the project limited its scope:

No Synthesis, No Sound Generation: A lot of research has been done on mapping gestures to musical parameters in the context of sound synthesis/generation, e.g.[4]. To the contrary for this project synthesis or any sound generation or playback triggered by the dancer was excluded from the very beginning. The role of the dancer should always be to modify the instrumental sounds through live electronics, but not to add additional acoustic events.

No Choreography: One of the other premises of the project was that the dancer should never follow any kind of choreography. In any moment of the performance the movements of the dancer should be musically functional, creating a clear visible relationship between the cause (motion) and the effect (sound modification). But any motion patterns from a choreographic repertoire should be abvoided. Therefore gesture recognition (e.g. through neural networks, [8]) is not required.

Therefore the mapping framework has to be seen as a playground for the dancer, giving him/her a certain influence over the distortion / alteration of the ensemble's sound. Practically the number of parallel mappings is (almost) unlimited, allowing one-to-one, one-to-many and many-to-many mapping connections [10].

As a first step we are calibrating the coordinates of the joints provided in centimeters (distance from the main camera) into an abstract range between 0 and 1 in order to make the musical effect of the dancer's motion independent from the actual size of the working volume on stage.

Secondly we generate derived parameters from the measured data: distance of hands, distance of feet, speed of left/right hand, speed of left/right foot, speed of head. These parameters are also calibrated into a range between 0 and 1.

In a following step these data streams can now be used in a general mapping module. The user selects a data stream being either a coordinate (x/y/z) of a skeleton joint or one of the derived parameters mentioned above. An output range is given to control the amount of the affected musical parameter.

Additionally there are two possibilities for non linear mapping: exponential/logarithmic mapping (figure 5)

$$y = x^a$$

where $(a, x \in \mathbb{R}, a > 0, x \in [0, 1])$, and s-shaped nonlinearity (figure 6) obtained by adding the result of the sine function to the linear/logarithmic/exponential mapping

$$y = x^a + b\sin(2\pi x)$$

where $(a, b, x \in \mathbb{R}, a > 0, b \in [-0.15, 0.15], x \in [0, 1])$. When using the position coordinates of the dancers on the floor (being x and z in the representation being sent from the sensors), it has proved to be important to have several mapping methods: a) positive/negative: in this case only half of the coordinates range is used. For example if the x coordinate is used in mode "positive", the mapping creates constant minimum values from the left side of the stage until the middle, and the musical parameter is scaled up to the maximum value only between the middle of stage and the right side. This method can be used to define areas in the corners of the stage, where e.g. most distortion only happens, if the dancer is moving out of the center of the stage into the extreme areas of the working volume. b) The opposite is possible in the "both centered" mode, where the maximum scaling is applied in the center of the stage, while the minimum output value is sent at both sides at the end of the stage.

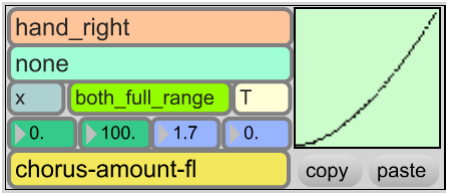

Figure 5: Exponential mapping of parameters.

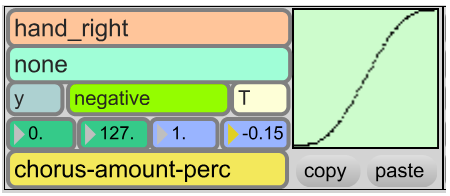

Figure 6: Sine mapping of parameters.

Finally the musical parameter to be affected is specified, e.g. volume amount of distortion effect, center frequency of filter, delay time in milliseconds etc. The system provides an arbitrary number of parallel parameter mappings, so that several aspects of the motion tracking and affect several musical parameters in parallel. Nevertheless practice has shown that more than 10 or 12 mappings in parallel have usually resulted in the fact that the audience could not perceive the interaction between movement and sound very well.

8. RESULTS

Using multiple Kinect sensors for tracking a dancer on stage is very stable and has proved reliable in a concert situation. Nevertheless certain constraints had to be observed:

The maximum distance between the sensors and the dancer was limited to approximately 3 to 4 meters, even when the stage light was reduced to 50% from the normal light level in the theater. Since the sensors need to project a grid of infrared light on the dancer, strong theater lighting "blinds" the sensors. Therefore a compromise between the visibility of the dancer and the possible size of the working volume had to be found. (See Figure 1). A *larger working volume* would also have been desirable, but would have required to dim the lights even further down, which was not acceptable for the audience.

The *tracking latency* is acceptable, usually a few tenths of a second.

The system has proved to be very robust. Re-calibration of the working volume for the merging software was only required, when the positioning of the sensors was unintentionally shifted by the participants. Otherwise the system runs stably even during several hours of intense rehearsing and/or performing. The leaving and reentering of the dancer to the working volume did not cause any tracking problems, since the Microsoft SDK re-initializes the tracking quickly and without special *calibration pose* requirements.

As can be seen in figure 4, our set-up uses one laptop per sensor. A three sensor system costs about $1.500 depending on the laptop used. A broken sensor – always a possibility

in a stage environment – is inexpensive ($100) and easily replaced.

So far three compositions have been realized with the presented system, commissioned by The XXth Century Ensemble Vienna:

- Wolfgang Liebhart: "Pas à pas"
- Se-Lien Chuang: "coincident synchrisis I, II, III, IV"
- Johannes Kretz: "Wortlose Räume III"

Nevertheless there is a wide range of compositions which were realized with earlier combinations of hard- and software. The experiences with the previous setups were essential to build the current system and a lot of knowledge and methods from there can be found in the current system, which provides both, a flexible, but consistent relationship between the dancers movements and the sounding effects as well as the freedom to use it in various aesthetic contexts.

The concert of The XXth Century Ensemble Vienna on the 9th Dec. 2012 at Palais Kabelwerk Vienna was a successful demonstration of the systems capabilities and maturity.

9. FUTURE WORK

For future projects it would be desirable to use stronger stage lights. It has to be examined if external custom hardware for projecting a stronger/ brighter grid of infrared light could be used to be able to augment the working volume and allow a brighter stage.

The Microsoft SDK in its current version does not allow to distinguish between front- and back-facing poses, which makes tracking e.g. the left or right hand difficult when the performer turns 180°. A heuristic to track the rotation of the performer has still to be implemented.

The use of sensor data to control lighting and visual projections will also be implemented in future versions of the project.

10. ACKNOWLEDGMENTS

This work has in been part funded by the Austrian Science Fund, Innovationsscheck #837700.

11. REFERENCES

[1] D. A. Butler, S. Izadi, O. Hilliges, D. Molyneaux, S. Hodges, and D. Kim. Shake'n'sense: reducing interference for overlapping structured light depth cameras. In *Proceedings of the 2012 ACM annual conference on Human Factors in Computing Systems*, CHI '12, pages 1933–1936, New York, NY, USA, 2012. ACM.

[2] C. C. Holt. Forecasting seasonals and trends by exponentially weighted moving averages. *International Journal of Forecasting*, 20(1):5 – 10, 2004.

[3] B. K. P. Horn. Closed-form solution of absolute orientation using unit quaternions. *J. Opt. Soc. Am. A*, 4(4):629–642, Apr 1987.

[4] A. Hunt, M. M. Wanderley, and M. Paradis. The importance of parameter mapping in electronic instrument design. In *Proceedings of the 2002 conference on New interfaces for musical expression*, NIME '02, pages 1–6, Singapore, Singapore, 2002. National University of Singapore.

[5] iPi Soft, LLC. iPi Motion Capture™ . `http://www.ipisoft.com/`. Accessed: 12/01/2013.

[6] A. Maimone and H. Fuchs. Reducing interference between multiple structured light depth sensors using motion. In *Virtual Reality Short Papers and Posters (VRW), 2012 IEEE*, pages 51 –54, march 2012.

[7] Microsoft. Microsoft Kinect™ for Windows. `http://www.microsoft.com/en-us/kinectforwindows/`. Accessed: 12/01/2013.

[8] P. Modler. Neural Networks for Mapping Hand Gestures to Sound Synthesis Parameters. In M. Wanderley, Battier, editor, *Trends in Gestural Control of Music*. IRCAM, Paris, 2000.

[9] T. Pintaric and H. Kaufmann. Affordable infrared-optical pose tracking for virtual and augmented reality. In G. Zachmann, editor, *IEEE VR Workshop on Trends and Issues in Tracking for Virtual Environments*, pages 44–51, Aachen, 2007. Shaker Verlag. Vortrag: IEEE Virtual Reality 2007, Charlotte, NC (USA); 2007-03-14 – 2007-03-17.

[10] J. C. Schacher. Motion To Gesture To Sound : Mapping For Interactive Dance. In *Proceedings of the International Conference on New Interfaces for Musical Expression*, pages 250–254, 2010.

[11] A. Schmeder, A. Freed, and D. Wessel. Best practices for open sound control. In *Linux Audio Conference*, Utrecht, NL, 01/05/2010 2010.

[12] A. Stelzer, K. Pourvoyeur, and A. Fischer. Concept and application of lpm - a novel 3-d local position measurement system. *Microwave Theory and Techniques, IEEE Transactions on*, 52(12):2664 – 2669, dec. 2004.

[13] G. Vigliensoni and M. M. Wanderley. A quantitative comparison of position trackers for the development of a touch-less musical interface. In G. Essl, B. Gillespie, M. Gurevich, and S. O'Modhrain, editors, *Proceedings of the International Conference on New Interfaces for Musical Expression (NIME)*, Ann Arbor, Michigan, May 21-23 2012. University of Michigan.

[14] A. D. Wilson and H. Benko. Combining multiple depth cameras and projectors for interactions on, above and between surfaces. In *Proceedings of the 23nd annual ACM symposium on User interface software and technology*, UIST '10, pages 273–282, New York, NY, USA, 2010. ACM.

[15] D. Zicarelli. An extensible real-time signal processing environment for max. In *Proceedings of the International Computer Music Conference (ICMC 1998)*, Ann Arbor, Michigan, 1998. MPublishing, University of Michigan Library.