Kinect-Via- :

Max/MSP Performance Interface Series for Kinect's User Tracking via OSC

Jon Bellona
University of Oregon
Digital Arts Department

bellona@uoregon.edu
http://deecerecords.com

ABSTRACT

In this paper, I describe a Max/MSP interface series (*Kinect-Via-*) for composers wanting to route and map user-tracking data from the XBox Kinect. The interface series complements four different OpenNI applications, namely OSCeleton, Synapse, Processing's simple-openni library, and Delicode's NIMate. All Max/MSP interfaces communicate using OSC (Open Sound Control) messages and are performance-ready, meaning that all routing and system options may be changed in real time. The *Kinect-Via-* interfaces offer a tangible solution for anyone wishing to explore user tracking with the Kinect for creative application. The aim of the paper is to discuss features of four different OpenNI applications, to address potential issues and challenges when working with the OpenNI framework, and to outline formative interface issues revolving around video tracking technology.

Keywords: XBox Kinect, OpenNI, Max/MSP, Live Interface, Open Sound Control, OSC, OSCeleton, Synapse, Processing, Delicode, NIMate, Kyma.

1. INTRODUCTION

The objectives of the *Kinect-Via-* interface series are threefold: to serve as a ready-made composition tool; to save users time in building a mapping framework between compositions; and, to act as a performance interface. Once Kinect drivers have been installed and the appropriate interface downloaded,[1] the *Kinect-Via-* interfaces handle all incoming OSC messages from respective software automatically.[2] The *Kinect-Via-* interface series provides ready-to-use data mapping objects inside Max/MSP, and each interface provides controls for communicating with their respective OpenNI application.

Before jumping into technical specifics, I will first contextually discuss the need for a Kinect interface. Next, I will analyze features of the OpenNI framework by highlighting four OpenNI applications. Third, I will discuss potential issues of video tracking through a real-world implementation of a *Kinect-Via-* interface. As a guide to the reader, all term definitions used throughout the paper may be found in section 8.

User tracking is a reality that offers many rewards, but not without its challenges. My hope is that the *Kinect-Via-* interface series hastens your discovery of new works and furthers your exploration of 3D space. At the time of this writing, the *Kinect-Via-* interface series has been downloaded over 1,000 times. [5]

2. THE INTERFACES

Anyone wishing to skip this article and jump right in with the open source interfaces may do so. The various interfaces– *Kinect-Via- OSCeleton, Synapse, Processing, NIMate* –may all be downloaded at: http://deecerecords.com/kinect

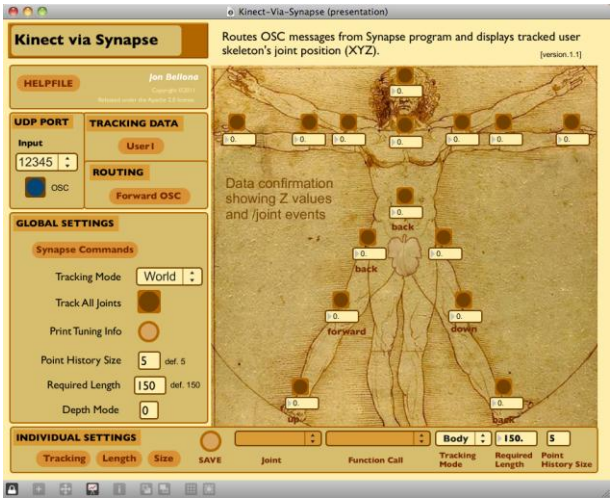

Figure 1. *Kinect-Via-Synapse* interface.

3. WHAT'S THE BIG DEAL?

Tracking users in space is not a new concept. David Rokeby's *Very Nervous System* tracked users with the computer as far back as 1982,[4] and many other video and light tracking systems have been introduced since.[3] What's stimulating about the Kinect is the affordability of tracking multiple users in 3D space, especially providing joint coordinates that may serve as real-time controllers. Ever since the initial Kinect 'hack,' [1] there have been hundreds of projects featuring Kinect user tracking,[4] and one-step installs have enabled quick

[1] All interfaces may be downloaded at: http://deecerecords.com/kinect
[2] I bundled a comprehensive sketch for use with Processing, while all other software automatically transmit Kinect tracking skeletons.

[3] Tracking systems like Imago, smart Junior, and EthoVision demonstrate available tracking systems, and software like Open CCV and Isadora take advantage of USB web-cameras.
[4] One of my favorite Kinect projects is Robert Hodgin's *Body Dysmorphia.* (http://roberthodgin.com/body-dysmorphia/)

access to the Kinect camera and the OpenNI framework. So, why a Kinect interface?

The *Kinect-Via-* interfaces address six major issues related to composition and performance process with the Kinect: data access; OpenNI feature access; standardized protocols; mapping framework; real-time options; and interaction design practice. With all the projects in existence demonstrating the Kinect's user tracking, there are few, open-source, modular interfaces to support a creative practice. In addition, provisions to access OpenNI features by applications are inconsistently offered or non-standardized, including the portability of data mapping modules, the controls for real-time options, and the availability of OpenNI features.

3.1. Plug n' Play (Data Access)

For anyone beginning to work with the Kinect, a creative starting point hardly exists. Even with knowledge of a programming language, many who work with the Kinect spend more time huddled over the keys of their computer keyboard than moving in the front of the Kinect camera. And, while most applications and libraries now have Kinect drivers bundled or single-click installers (i.e. OSCeleton, Processing's simple-openni, Synapse, NIMate), an interface isn't provided for parsing incoming OSC messages. In fact, most system options are not available unless users create a bi-directional OSC architecture for the parent software. The *Kinect-Via-* interfaces serve as this missing communication framework, saving time in the creation of necessary application components. The interfaces provide immediate data access without having to program a single line of code.

3.2. Extending OpenNI Feature Access

A frayed thread of continuity exists between OpenNI applications and the OpenNI framework, as each OpenNI application selectively chooses features from the framework to support ad hoc. One application will allow NiTE hand gesture tracking, while another will not. One application will transmit CoM data, while another application will not.

Add in a composition's requirements and designing a flexible system becomes a challenge. While one composition may call for one feature, say multi-user tracking, the next piece may demand different OpenNI features. Depending upon the application's available features and upon the composition's requirements, a user may be required to jump between applications or translate data from one application to another.

One solution to feature access is data transference. As each OpenNI application offers different OpenNI features, being able to port, or transfer, user generated mappings between applications enables access to more features of the OpenNI framework. *Kinect-Via-* interfaces keep all data mapping objects separate from the system features of each OpenNI application, so that custom mappings may be shared between any of the four different OpenNI applications with a simple cut & paste.

3.3. OSC Protocol

Using a stable, standard protocol for receiving messages provides a sustainable and flexible architecture, and all *Kinect-Via-* interfaces were built for applications that support OSC. The four OpenNI applications (OSCeleton, Synapse, Processing's simple openni, and NIMate) transmit OpenNI data (e.g user skeletons) as OSC messages. Some applications even allow options to be set with OSC. If offered by the OpenNI application, options controls are provided through the *Kinect-Via-* interface.

3.4. Mapping Framework

The *Kinect-Via-* interface series provides out-of-the-box mapping capability, as all user-tracking OSC messages received by Max are first unpacked into floating point numbers and then connected to accessible 'send' objects. Max 'receive' objects are all that are required to start mapping joint data. Data 'send' object names are shared between interfaces, so compositional mappings may be shared between OpenNI applications.

3.5. Real-Time Options

The *Kinect-Via-* interface series enables real-time options controls for the four OpenNI applications. For example, the *Kinect-Via-Synapse* interface provides global and individual controls for all options available to Synapse, without which a user would have to code his/her own OSC-enabled control panel.

3.6. Interface Design

Lastly, all *Kinect-Via-* interfaces maintain similar design scheme. The top level displays performance controls. The major design decisions– the Vitruvian man, interface controls, help files, even OSC parsing– remain consistent between each *Kinect-Via-* interface to decrease learn-time and to increase familiarity. Max 'send' objects are the same for each *Kinect-Via-* interface so that any artist can reuse Max abstractions and patches between applications.

4. SYSTEM FEATURES

This section outlines the main features of the OpenNI framework by highlighting the similarities and differences between four OpenNI applications (OSCeleton, Synapse, Processing's simple-openni, NIMate).

4.1. User IDs and CoM (Center of Mass)

As soon as a user is identified in the tracking space, the OpenNI framework assigns the user an ID number, numerically incrementing from 1-16. This unique user ID follows the user throughout the space, and remains logged even when a user leaves the Kinect's field of vision. In fact, the user ID remains logged for ten seconds, and currently, this log time remains an unchangeable OpenNI constant.

Because the OpenNI framework dynamically assigns user IDs, a user cannot guarantee that he/she will have the same user ID number for calibration. As

such, explicitly dictating a user ID for mapping without including re-routing controls should be avoided. All interfaces address this issue– *Kinect-Via-OSCeleton* and *-NIMate* offers re-routing of user IDs in real time, while the *Kinect-Via-Synapse* and *-Processing* interfaces dynamically switch between user IDs.

In addition to user ID assignment, OpenNI reports the center of a user's body (x,y,z), known colloquially as CoM (Center of Mass). CoM does not require calibration. Not all OpenNI applications transmit CoM data, however. All applications except Synapse make CoM data accessible.

4.2. The Psi Pose and Auto-Calibration

The OpenNI framework must calibrate a user before transmitting skeleton coordinates.[5] Prior to 2012, OpenNI applications required a user to pose in order to initialize calibration.[6]

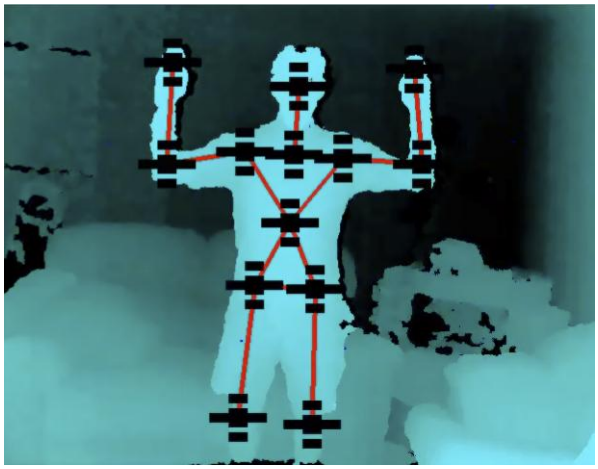

Figure 2. A calibration process using the "Psi Pose."

Now, many OpenNI applications offer auto-calibration. Auto-calibration calibrates a user automatically, and the process starts as soon as a user receives a user ID. In addition, a user is not required to stand still. Auto-calibration is a great feature for any user interested in not having to issue a pose in order to implement joint tracking. Auto-calibration may not be useful for public art performances, where unwanted user-calibration can occur.

Between user calibration and skeleton data transmission, a lag time exists.[7] For performance, lost data as a result of calibration lag time may be unwanted, but user IDs can serve as a stopgap. By adding a check against the tracking skeleton user ID and other existing user IDs, one may switch between user skeletons without requiring re-calibration. This check allows for a seamless (non-latent) transition of tracking joints between users.[8]

In addition to auto-calibration, Processing and NIMate offer the ability to save and load different calibration poses. The need for working with different calibration poses might serve multiple users requiring a skeleton at different times, especially when multiple users exist within the tracking space.

4.3. Single-User and Multi-User Tracking

The OpenNI framework supports multi-user tracking, where multiple skeletons may be present and transmitted at one time. Yet, there is no standard. Applications choose to support single-user tracking (access to only one skeleton), multi-user tracking, or both. Synapse offers single-user, OSCeleton multi-user, and both NIMate and Processing offer both.[9]

Tracking multiple, moving users has potential issues. Depending on the position of the Kinect camera, multiple users inside the space conceal tracking joints, which may result in lost user IDs. Reliably tracking more than two users, especially multiple users who move around and exit the tracking space, will require additional conditional coding to handle lost joint data, lost user IDs, and the ten second OpenNI constant that logs user IDs.[10]

The most reliable results for accessing and mapping skeleton joint data involve one-to-two users inside the tracking space at any given time. Due to multi-user mapping challenges and the stability of single-user tracking, digital ensembles should strongly consider a maximum of one-two users per computer/mapping interface.

4.4. Tracking Modes

Tracking modes enable different coordinate data sets for a user's joints, offering flexibility in data manipulation. While the OpenNI framework offers various tracking modes, including user-defined modes, not every OpenNI application takes advantage of these modes.

4.4.1. Real-World vs. Projective[11]

As best described by Greg Borenstein, "Real-world coordinates locate the object in 3D space. Projective coordinates describe where you'd see it on the viewing plane." [2] NIMate and Processing offer both real-world and projective modes, and both are changeable in real time. The simple-openni Processing library only presents examples in real-world mode, so in order to harness the rich sets of data that tracking modes offer, I scripted a Processing sketch to switch between real-

[5] There are fifteen joints currently supported by OpenNI as part of a user skeleton, even though more joints have been programmed into the framework. The currently accessible joints are head, neck, torso, left hand, left elbow, left shoulder, right hand, right elbow, right shoulder, left hip, left knee, left foot, right hip, right knee, and right foot.

[6] The most familiar pose associated with the Kinect is the 'psi pose,' named for its relationship to the Greek letter Psi.

[7] The lag time is around two seconds, reported from tests across three applications supporting the auto-calibration feature.

[8] Switching user skeletons in this fashion was tested with three users using Processing. The feature will be part of an upcoming work, *Maia*, by Harmonic Lab and will involve three dancers sharing one tracking skeleton. http://harmoniclab.org

[9] While Processing's simple-openni library offers the capability to track multiple skeletons, the Processing sketch complementing the *Kinect-Via-Processing* interface currently offers only single-user tracking.

[10] See Section 5.1 for one solution.

[11] Some applications choose to use the term "Screen" mode. "Projective" mode is synonymous with "Screen" mode.

mode and projective modes, controlled by the *Kinect-Via-Processing* interface.

4.4.2. Body Mode

Synapse is the most robust application that handles tracking modes, offering real-world, projective, and a third, unique mode– body. Body mode sends coordinates as joint distance measurements relative to the torso, expressed in milli-meters. All three of Synapse's modes may be switched globally or on a joint-by-joint basis, meaning that individual joints may send coordinates in different modes.

4.4.3. Coordinate Values

Coordinate set values vary between modes. While projective mode scales the range of X values to pixel screen width (0-640), the range of X values for real-world and body tracking modes are expressed in negative and positive milli-meter values. The zero point corresponds to the center point of the Kinect camera. The same is true for Synapse's body mode, except that the zero point always moves relative to the X position of the torso.

4.5. NiTE Gestures and Joint Triggers

NiTE is one middleware that complements the OpenNI framework, providing the technological guts of gesture-based control, where specific gestures may become actuators[12]. Processing offers the most flexibility in harnessing NiTE gestures; however, a comfortable understanding of programming is necessary, as one will need to code in the supporting functions. NIMate capitalizes on NiTE by allowing users to switch between skeletons by using a hand wave gesture.

Gestures may be mapped even without NiTE functions. Synapse offers actuation messages (back-forward-left-right) for joints moving directionally in space. Both the velocity and length required to trigger these messages may be altered in real time.

4.6. Real-time Options

A performance interface with real-time options has major benefits, notably the ability to change system behaviour, quickly test ideas, and resolve unexpected issues on-the-fly.

NIMate and Synapse make the most out of providing the user with real-time options. NIMate offers these options as part of its application, while Synapse only provides the framework. *Kinect-Via-Synapse* provides ready-made access to getting and setting Synapse's user-tracking options, and with both global and individual joint settings controls.

[12] 'Actuators' here mainly refers to Wanderley's usage in *New Musical Digital Instruments,* [6] but instead of electrical energy, the human body's mechanical energy is converted into digital bits used to propel a mapped action. The actuator in the digital form may be a trigger, boolean, or gate.

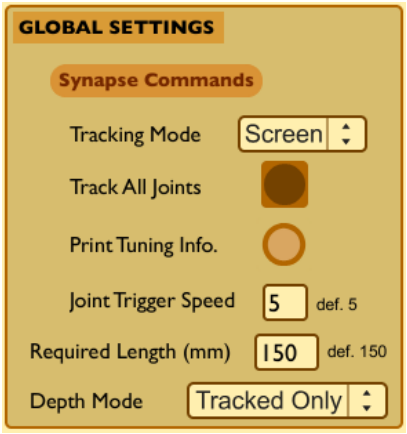

Figure 3. The *Kinect-Via-Synapse* global options panel.

4.6.1. Synapse

Kinect-Via-Synapse makes available these real-time options: tracking mode (body, real-world, projective); track joints on/off toggle (including individual joint tracking switches to help save CPU); print tuning info (prints current joint settings); joint trigger speed (joint trigger velocity threshold); required length (joint trigger distance threshold), and depth mode (alternate depth views from the Kinect camera).

4.6.2. NIMate

NIMate offers the most real-time options, but it is important to note that any option change will cause the system to re-calibrate the user. In NIMate's defense, the application allows user profiles to be saved and loaded, ensuring a performance-ready system. Real-time options available to NIMate include tracking mode (real-world, projective), joint smoothing, a scalable activation area, joint mirroring, coordinate scaling, OSC output formatting, and MIDI controller output.

4.6.3. OSCeleton

While OSCeleton does not offer any real-time options, the *Kinect-Via-OSCeleton* interface enables the re-routing of user IDs in real time. By transferring the routing of user IDs to the Max interface, the user gains mapping flexibility over parsed OSC joint messages and ensures that OSCeleton will not have to rebooted during a performance due to unmatched user IDs.

4.6.4. Processing

Because Processing is a programming environment and not a closed application, options can be added and modified at will; however, the simple-openni library does not initially provide real-time controls. In order to provide real-time options for performance, I bundled a Processing sketch with the *Kinect-Via-Processing* interface that enables several real-time options: the ability to change tracking modes, a toggle for getting CoM and skeleton joint data, and getting the distance in milli-meters between hand joints.

5. REAL-WORLD EXAMPLE: *HUMAN CHIMES*

One example of the *Kinect-Via-* interface in use is the interactive sound installation, *Human Chimes*. *Human Chimes* dynamically tracks users' locations (CoM) in real time with OSCeleton, and the piece maps participants as sounds that pan around the space according to the participants' positions. Sounds bounce between all other participants inside the space, and a Processing sketch provides user feedback by projecting participants' movements onto the front wall.

Figure 4. *Human Chimes* technical layout: Kinect on a telescoping microphone stand, projector, MacBook displaying depth mode screen, and Kyma system.

Figure 5. *Human Chimes* installation. White balls represent users in the space, which also control the panning location of emitted sounds.

Several challenges arose while working on *Human Chimes*, and highlighting three here may help outline issues of working with the OpenNI framework.

5.1. Issue #1: Lost User IDs

Because users entered and exited the installation space at random, there was high risk for lost user IDs and 'stuck' user CoM coordinates.[13] To resolve this issue, I added a validation check of user CoM coordinates. The CoM coordinate (or torso coordinate for calibrated users) equals (0.0, 0.0, 0.0) when a user is lost, even though the user ID remains logged. Saving user IDs into a second, conditional-based array for mapping user coordinates ensured only active IDs would be mapped. Validating user IDs through a CoM coordinate check is one method for working around the OpenNI constant and lost user IDs.

5.2. Issue #2: Projecting Onto The Tracking Space

Transferring the tracking range of the Kinect to a projection, either onto a wall or onto the tracking space can be difficult. Projecting objects back onto the tracking space requires three levels of scaling. First, real-world X coordinates must be scaled to the tracking space projection. Second, these coordinates must be scaled again to the computer display screen width. Lastly, Z coordinate must be scaled to the computer display screen height.

One useful tool in mapping user location to projection location is to create a real-world mapping box. By demarcating the X and Z coordinate values of real-world coordinates at the edges of the projection inside the tracking space, and by setting these coordinates as a global variable, one can, in effect, create an invisible bounding box. The bounding box may be used to map coordinates back onto a projection covering the tracking space. The box also enables performance by allowing the range to be easily reset for any given space.[14]

5.3. Issue #3: Limited Tracking Range

While the Kinect supports tracking to about 26 feet (8000 mm), the stable, accurate tracking range is much more limited. The reliable, accurate tracking range for skeleton joints and CoM data reflects a maximum depth of about 15 feet.[15] While I have no work around to offer, understanding this limit is helpful when programming for music/dance performance or installation art.

Every composition faces its own set of technical hurdles, and hopefully, the ones addressed here present possible solutions for working with the Kinect and OpenNI framework. For more on *Human Chimes* and to view the installation documentation, please visit: http://deecerecords.com/projects#humanchimes

6. AND THE WINNER IS!...

...Well, it depends. Because of the robust complexity of the OpenNI framework, each application offers different features. Additionally, every Kinect user has different needs. To help guide the reader, I have provided a feature list below, a supplement to sorting out the various applications. For each case, the applications are listed in ranking order from left to right, from most to least useful.[16]

[13] As previously mentioned, the OpenNI logs user IDs for ten seconds after a user leaves the tracking space.

[14] One can easily map the tracking range, regardless of the performance space, by running the OpenNI application and notating a user's CoM or joint torso coordinates as he/she moves around the space.

[15] The distance is based upon location measurement tests, including *Human Chimes* installation at three different locales, practice tests, and other interactive, Kinect-based work.

[16] The phrase 'most to least' is based upon the experience and the bias of the author.

N=NIMate, O = OSCeleton, P=Processing, S=Synapse

FEATURE	1	2	3	4
Multi-user skeleton tracking	O	N	P*	
Auto-calibration support	P	N	O	
Tracking mode support	N	S	P	
Relative joint positions	S	N	P*	
CoM coordinates	O	P	N	
Real-time options support	S	N	P*	O
Software reboot time**	O	S	N	P
Installation process	S	N	P	
NiTE Gesture support	P*	N		
low CPU usage***	S	O	P	N
* possible, but requires additional coding				
** O: 7-10 sec., S: 10-12 sec., N: @30 sec., P: @30 sec.				
*** based on Activity Monitor of MacBook Pro (10.6.8)				

Table 1. OpenNI feature chart, showing the various application support of OpenNI features.

7. CONCLUSION

The OpenNI applications I chose to work with are not an exhaustive list, and there are certain limitations in creating a second-party interface, most notably the segmentation of OpenNI features. Making the most out of the OpenNI framework and its features is an ongoing process, one that develops alongside the framework. The Kinect is not quite two years old, but already has proven to be an exciting tool with tremendous possibilities for the performance and interactive arts.

Lastly, the *Kinect-Via-* interface series is a tool. The interfaces provide the ability to start composing immediately while delivering instant access to real-time controls. The series was made with composition and performance in mind, as well as for any user-level, including those with little to no programming knowledge. More experienced users may find the modular interface time-saving, and especially useful for experimentation and programming hacks.

Once again, the various interfaces– *Kinect-Via-OSCeleton, Synapse, Processing, NIMate* –may all be downloaded at: http://deecerecords.com/kinect

8. DEFINITIONS

This section outlines terms used throughout the paper, and each term builds off of previously used terms.

Kinect - The Kinect is a motion sensing device for the XBox 360. Yet, the Kinect is not just a camera nor a gaming experience, but a revolution. The Kinect device enables a new world of possibilities, including 3D computer vision, user-tracking, gesture analysis, and even 3D coordinate manipulation in real time.

OpenNI - OpenNI is "a framework that provides an application programming interface (API) for writing applications using natural interaction,"[9] and in our case, that application is the Kinect. OpenNI allows us to track users in space, grab specific coordinate sets, as well as extend the existing framework with 'middleware' software (e.g. NiTE gesture support).

Middleware - Middleware is computer software that connects various software components with other applications, thereby extending an application by enabling multiple processes running on one or more machines to interact.[3] NiTE is one such middleware for the OpenNI framework.

NiTE™ (Primesense's Natural Interaction middleware) - NiTE is a middleware for OpenNI containing the "algorithmic infrastructure for user identification and gesture recognition, as well as the control framework that manages the tagging of users and the acquisition and release of control between users." [8]

OSC (Open Sound Control) - OSC is a stable, 32-bit protocol used for interconnecting hardware controller devices to the computer, as well as software on one or more computers. The OSC protocol was developed by Adrian Freed and Matt Wright in 2002 at CNMAT (Center for New Music and Audio Technologies), and the protocol utilizes UDP/IP (User Datagram Protocol/Internet Protocol) packets, which are user-defined packets of information sent to/from computers and devices on the same local network. [7]

Track - Through the analysis of points collected from IR lights, the Kinect can distinguish between objects in 3D space. To track means that a moving body in space receives a user ID, and whose collection of pixels is identifiable and separable from all other objects.

Tracking Space - The area in which a user is detected by the Kinect.

CoM (Center of Mass) - A user's densest point of the body, the torso, is sent as a 3D coordinate. This feature is enabled as soon as the Kinect detects a moving body in space. Note: In testing motion, a chair rolling across the tracking space may feasibly send a CoM coordinate.

Calibration - The OpenNI framework must calibrate a user before specific joint coordinates can be collected and transmitted. Calibration may be handled automatically, or through an identified pose, like the 'psi pose.' Note: CoM data does not require calibration, and CoM is the same as a calibrated torso joint.

User/Tracking Skeleton - A calibrated user shows and sends up to fifteen joint coordinates. These coordinates may be sent in a variety of different modes (real-world, projective, body). The number of skeletons supported is dependent on the OpenNI application, but a maximum of sixteen user IDs are supported by the OpenNI framework.

9. REFERENCES

[1] Adafruit. 2010. We Have a Winner: Open Kinect driver(s) released. *Adafruit Industries Blog* (Nov. 11, 2010). DOI= http://www.adafruit.com/blog/2010/11/10/we-have-a-winner-open-kinect-drivers-released-winner-will-use-3k-for-more-hacking-plus-an-additional-2k-goes-to-the-eff/.

[2] Borenstein, G. *Making Things See.* Sebastopol, USA, 2012, 204.

[3] Campbell, A., Coulson G., and Kounavis M. "Managing Complexity: Middleware Explained." *IT Professional, IEEE Computer Society*, 1:5, September/October 1999, 22−28.

[4] Cooper, D. "Very Nervous System," *Wired Online.* http://www.wired.com/wired/archive/3.03/rokeby.html last accessed August 3, 2012.

[5] "http://deecerecords.com," *Google Analytics*. http://google.com/analytics August 1, 2012.

[6] Miranda E., and Wanderley M. *New Musical Digital Instruments: Control and Interaction Beyond the Keyboard*. Middleton, WI., 2006, 104.

[7] Open Sound Control. http://opensoundcontrol.org/introduction-osc August 3, 2012.

[8] Primesense™, Inc. NiTE Middleware. https://www.futureworld.sg/services/naturalinteraction/nite August 8, 2012.

[9] Primesense™, Inc. *PrimeSense NiTE Controls 1.3.1 User Guide*. March 2011.