LeapMax: Gestural Interaction System

George Cooper Jones
Arts, Media, and Engineering
Arizona State University
gcjones1@asu.edu

ABSTRACT

The LeapMax Gestural Interaction System is a project which utilizes the Leap Motion controller and visual programming language Max to extract complex and accurate skeletal hand tracking data from a performer in a global 3-D context. The goal of this project was to develop a simple and efficient architecture for designing dynamic and compelling digital gestural interfaces. At the core of this work is a Max external object for extracting data from the Leap Motion service. From this, a library for determining more complex gesture and posture information was generated and refined. To demonstrate the use of this system in a performance context, an experimental musical instrument was designed in which the Leap is combined with an absolute orientation sensor and mounted on the head of a performer.

1. INTRODUCTION

1.1 Glove Based Interactive Hand Tracking Systems

The proliferation of inexpensive and highly accurate wearable sensing technologies has resulted in the widespread development of wearable devices and peripherals in the fields of human-computer interaction, digital musical instrument (DMI) design, and beyond. These devices can be used to develop intuitive and highly dynamic interaction methods by developing relationships between the physical state of the subject being measured and the values output by a sensor or sensors. Measuring the human hand is of particular interest in this respect due to the complexity of movement and expressivity that it possesses. Hands are the means of interaction for almost all musical instruments, so it stands to reason that a lot can be gained from tracking and mapping their gesture and posture.

One of the earliest examples of hand tracking technologies is the Nintendo Power Glove. The Power Glove was a peripheral game controller release in 1989 for the Nintendo Entertainment System. It allowed for basic game interaction and control based on the bend of the fingers, the hand's orientation in space, and an assortment of buttons placed on the forearm of the glove [17]. Although it was not very successful, it represents one of the first examples of a data glove system. A later and far more sophisticated example of a data glove system is Laetitia Sonami's Lady's Glove. This device is a gestural musical interface debuted in 1991 that, throughout its evolution over 25 years, used a variety of hall effect sensors, ultrasonic sensors, embedded switches, bend sensors, and pressure pads to detect movement and gesture and transform it into sound [11]. Other examples of more modern data glove systems include the Mi.Mu glove system and the Alto.Glove system by Seth Thorn [4, 15].

The technology described in this report is not meant to be a competitor to the current methodologies of glove and sensor-based hand tracking technologies, but rather it is meant as a complementary approach to capturing posture and gesture data from hands which presents its own advantages and disadvantages when used in different contexts. These advantages and disadvantages will be discussed later in the report.

1.2 Leap Motion

The Leap Motion is a small rectangular device which uses infrared cameras to calculate and track the positions and postures of hands in front of the sensor. It has been used extensively in the field of digital instrument design and human computer interaction. The Leap Motion was first released in 2013 and has since gone through several iterations of development which have each shaped and changed its goals, purpose, and uses. When it was first released, the Leap was used mainly as a desktop device which could detect the position and rotation of a hand as well as gather limited finger measurements. This system was very good at tracking the orientation and movement of the whole hand, but limitations in the software's ability to track fingers meant that minute motions and posture were almost impossible to accurately measure [1]. Early gestural music software for the Leap includes Geco Midi [7], a software for converting hand positions and rotations into midi values for parameter control, AirHarp [2], a digital harp controlled by the leap, and Flocking [6], a musical simulation of a school of fish that react to the interactor's hand.

In 2016, the Leap Motion team released the Orion software update which truly revolutionized what the Leap device was capable of. This software focused on integrating the Leap Motion into virtual reality setups by adding new software configurations for a head-mounted setup and vastly improving the latency, reliability, and accuracy of the device. With this update, the Leap Motion now has two modes - one for a desktop Leap setup, and one for a head mounted Leap setup. Since then, the development path has focused heavily on improving the VR/AR paradigm, with the goal of replacing standard VR controllers with a more natural and gesturally interactive system.

In early 2018, the Leap 4.0 software was released, featuring a new streamlined C API and boasting better fidelity and consistency than ever before. The Leap C API has been released without any supporting language wrappers, with the hope that language wrappers will be developed by the community.

The Leap Motion, and other IR technologies like it, offer unique opportunities in the field of DMI design and gestural interaction. Whereas previous systems that measure hand gestures and movements have required the use of physical sensors to function as a layer of abstraction between the actual positioning of the hand and the data collected, the Leap Motion

has the unique capability of allowing direct access to the exact skeletal positioning of the human hand. No longer is the hand simply an actuator for some other dynamic system - be that a physical instrument, object, or sensor. Instead of being the method of control, the hands themselves can become the system to be measured.

2. BACKGROUND

2.1 Max + Leap Motion

Max (Max/MSP) is a visual programming language that puts an emphasis on music and multimedia and is widely used in the spheres of experimental sound, DMI design, installation art, and others. Because of its modularity and graphical user interface, it has been called the "lingua franca for practitioners of computer-based live performance" [18].

Despite the close relationship the Leap Motion has had with music and sound, an officially supported API for communicating between the Leap Motion and Max has never been developed. There are, however, two popular custom built APIs for Max and Leap: the aka.leapmotion object which was released in 2013 by Masayuki Akamatsu, and leapmotion-for-max which was released in 2014 by Jules Francoise [3, 13].

Akamatsu's object was developed very early on in the Leap's life cycle and, despite being heavily used during that time, it is now deprecated and no longer functional with later iterations of the Leap software. During this period of the Leap's development, the sensor did not label finger types or tell left hand from right, which became a frustration for many developers using the Leap with Max [9, 10]. Luckily, this is a problem that has been resolved by later iterations of the software, and the Leap service now identifies hand and finger types natively.

Francoise's object has received consistent updates which allow it to be functional with the Leap software up to version 3.0+. This object uses the deprecated C++ Leap API, however, and thus no longer functions with the Leap C 4.0 software.

One of the goals of the LeapMax project is to develop an updated Max API for communicating with the new Leap service protocol as well as create a library of objects for extracting more complex gestural data from the Leap Motion. The hope of the author is that by developing an efficient and straightforward Max protocol for the Leap, this technology will be made accessible to new groups of developers, musicians, and artists.

2.2 LeapMax and The VR Hand Tracking Paradigm

Along with developing a Max API and library for the Leap, another goal of this project is to demonstrate a use case of the system as a way to unite the head-mounted Leap VR concept with the methodologies used by data glove systems. Many virtual reality projects explore the applications of the Leap in audio design such as LyraVR, an interaction system that allows the user to create sounds and music using a variety of virtual objects [14]. Most of these systems focus on using the Leap's high fidelity tracking to fluidly interact with a simulated 3D environment which in turn produces sound. The LeapMax project removes the visual component of the VR experience, focusing instead on measuring the user's posture and gesture to form a unique gestural space within the context of the interactor's physical surroundings.

3. THE LEAPMAX API

The first portion of the LeapMax project was developing an updated and highly efficient Max API for communicating with the Leap service. This was done by developing a custom Max C external object which utilizes a combination of the Leap 4.0 C library and custom code to transform Leap data and send it to Max for use. The end result is a Max object called "leapmax" which accepts into its first inlet a bang which triggers the collection of a frame of data from the Leap service and saves it to a Max dictionary.

3.1 Previous API Designs and Considerations

The goal of this project is not only to update current Max APIs, but to improve the efficiency and cohesion of the system as a whole. This began with an examination of the previous API designs and their successes and shortcomings. Because of the age of the aka.leapmotion Max object, Francoise's leapmotion-for-max object was the main focus of this examination.

Leapmotion-for-max is a Max C external which accesses the Leap service through the Leap C++ API and then parses the Leap data and passes the resulting values as prefixed messages through the outlets of the leapmotion object. Four outlets send out gesture data, hand data, finger data, and frame data, respectively. Within Max, desired data can then be routed and extracted for use as necessary. Overall this system is well designed and implemented, but there are several areas where there is room for improvement and optimization.

One of these areas is the parsing and collection of particular messages containing data for use. Francoise's leapmotion object outputs 38 Max messages per frame, and all of these messages must be sent to every router object where a value is parsed. This quickly becomes a bottleneck when many values are being used by the system because of the quantity of Max messages being sent on every frame. Another issue presented by the current system is that the structuring of the leapmotion message data makes programmatically selecting certain values (ie. through a data collection abstraction) unintuitive. The LeapMax system resolves both of these problems by utilizing a Max feature called 'dictionaries'.

3.2 Max Dictionaries

Max version 6 saw the release of a new type of data structure called a 'dictionary'. Max dictionaries focus on organizing and efficiently passing structured data in a global scope. Rather than passing by value as standard Max messages do, dictionaries offer the capability to pass by reference which allows different objects to access data from the same memory space. This makes dictionary data structures a perfect candidate to hold data passed from the Leap because it is possible to update one single collection of data and retrieve values from that collection from anywhere in Max. This has the benefit of separating the data input method and the data retrieval method while also increasing the performance of the system as a whole.

3.3 Data Structure and Naming

In order to make the data in the Leap dictionary easily accessible, a consistent and modular naming system was needed. Initially, the data in the Leap dictionary was intended to be hierarchical, with a parent dictionary representing hands, subdictionaries within the hand dictionaries holding finger data, and subdictionaries within the finger dictionaries holding bone data. Although this makes sense from the standpoint of the structure of the data being received from Leap, in practice, the hierarchical structure ultimately added additional unnecessary complexity when accessing values, especially from different layers of the hierarchy.

Instead, the LeapMax API uses a single layer dictionary with a hierarchy built into the naming system that allows for an equally modular design which is far less complex. The LeapMax naming system corresponds to the data structure used by the Leap software, which structures data in four encapsulated classes: global, hand, finger, and bone. The global object contains two instances of the hand class, each of which contains five instances of the finger class, each of which contains four instances of the bone class. In the LeapMax naming system, data is labeled corresponding to its variable name prepended with a type at each layer of the hierarchy in which it is encapsulated.

For example, dictionary values in the global layer are accessed using just their variable names (e.g. *frameid* or *timestamp*), values in the hand layer are named by prepending the datas' variable names with the hand type being accessed (ie. *rightpinchdistance* or *leftgrabstrength*), values in the finger layer are named by prepending the variable name with the hand type and finger type (e.g. *rightindextipposition* or *leftthumbisextended*), and so on with the bone layer. The diagram below illustrates the structure of the naming system. This naming system not only makes accessing Leap dictionary data in Max very straightforward but also goes a long way toward the extensibility and modularity of the Leap library which will be discussed later.

system. In the desktop setup, the Leap device does not move and so this data is consistent with the global frame. In order to use the Leap in a head-mounted setup, the orientation of the device must be tracked so that the hand positions can be transformed accordingly in the global frame. The Leap documentation has extensive explanations of the matrix transforms required to

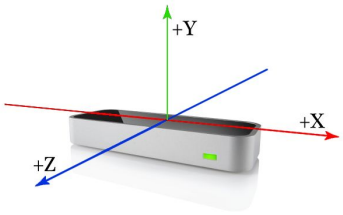

Fig. 2 [19]

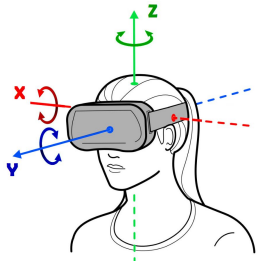

Fig. 3 [20]

convert the Leap coordinate system into global space [12].

In most VR setups both the orientation and the position of the headset are tracked and thus this data can be passed and used to transform the Leap data. There is no visual component of the LeapMax system, and thus, the absolute position of the sensor becomes irrelevant because only the relative positions of the hands from the head of the performer need to be measured. The Leap data only needs to be transformed by the rotation matrix representing the orientation of the device in order to extract useful data. Because the Leap data is only transformed by the device's orientation and not by position, the resulting data exists in what could be referred to as a 'pseudo-global' frame of reference. In this "pseudo-global" frame, the Leap data is oriented properly to the global frame, but the origin of the coordinate system is always the position of the leap sensor itself rather than some arbitrary static point in space. This has the added benefit of allowing the user to move/walk around in space without affecting the output of the system.

To accommodate the transformation of the Leap data from the Leap coordinate system to "pseudo-global" coordinate

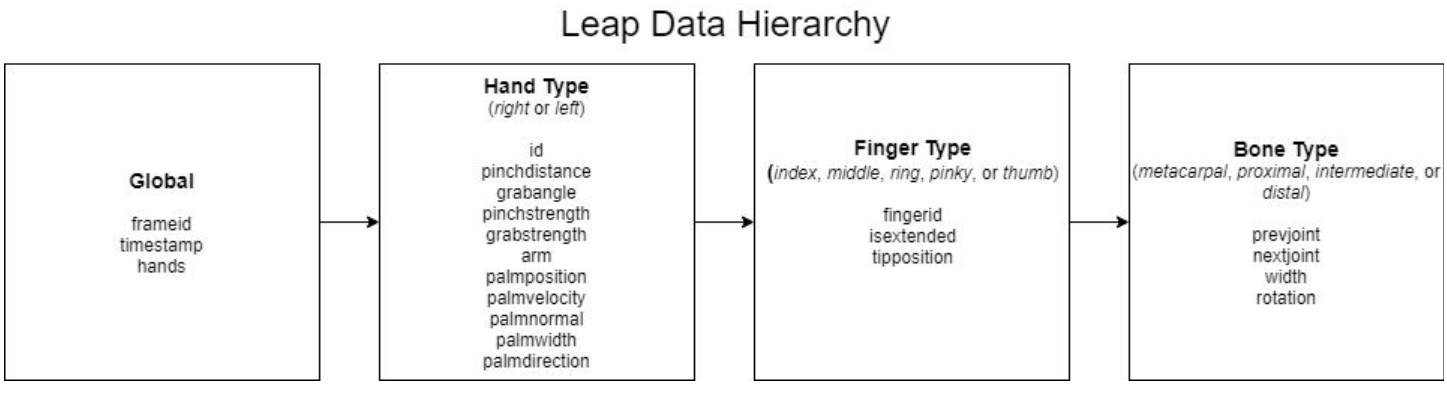

Fig. 1

3.4 Coordinate Systems and Orientation Management

The Leap Motion uses a right-handed Cartesian coordinate system measured in millimeters with an origin at the center of the Leap sensor device, as shown in Figure 2. On its own, the Leap returns data only with respect to this coordinate space, additional inlets were added to the leapmax object to accept orientation data for the Leap device in the form of a rotation quaternion. Each position value collected from the Leap service is then transformed by this orientation before it is output to the dictionary. A discussion of the methods and technologies used to track the orientation of the Leap device will be provided

later in Section 5, which outlines the LeapMax performance system.

4. THE LEAPMAX LIBRARY

The second portion of this project was the development of a library of objects that can be used to extract data from the Leap dictionary as well as extrapolate more complex data regarding posture and gesture. Figure 4 shows a list of all of the currently developed objects in the leap library. The goal here is to make designing and developing more complex systems using the Leap easier and more direct by leveraging a modular approach to object design which allows for a high level of reusability and extensibility. The most basic object in the leap library is leap.get, which, when given an argument matching the name of a dictionary key, returns the value(s) at that key. This object can be used on its own or encapsulated within more complex abstractions such as leap.velocity in order to collect and manipulate certain data in more complex ways. The following sections describe design decisions which ensure the flexibility and modularity of the Leap library as well as considerations regarding categorizing different types of measurable hand gestures and postures.

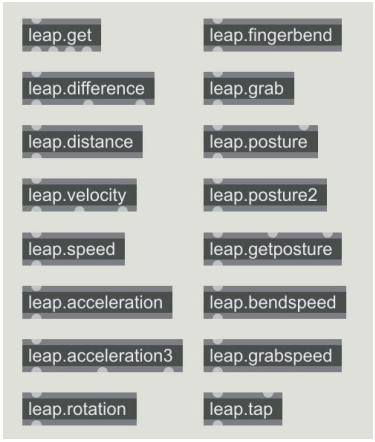

Fig. 4

4.1 Object Construction - Designing For Modularity

Modularity was an extremely important design focus of the Leap library. This is because to make the Leap library a highly effective tool, it was necessary to design a system in which the chosen tracking data could be easily specified for each instance of a Leap abstraction—should, for example, the rightindextipposition be tracked, or the leftpalmposition? It was also important to ensure that abstractions could be encapsulated in order to develop more complex gesture systems.

4.1.1 Scheduling

Because the data capture and data selection operations have been decoupled, capture operations need to be scheduled so that values only update when a new frame is read from the Leap service. This ensures that values are always up-to-date and stale values are never passed. Like the leapmax API object, each object in the Leap library has a left inlet which accepts a bang to trigger accessing data from the Leap dictionary. By connecting this to the same global metro that triggers the capture of a Leap frame in the leapmax object, all of the leap library objects can be synced with the global frame control.

4.1.2 Max Object Arguments

Max object arguments are additional parameters added after the abstraction name that can be passed to the Max abstraction and used anywhere in the encapsulated patch. Each of the Leap library objects accepts one or more arguments to determine what data they should measure. For example, if a user wants to measure the speed of the right index fingertip, they would use the leap.speed object with an argument "rightindextipposition." Arguments can be passed down and used in multiple layers of encapsulation within abstractions. In the example just given, the leap.speed abstraction passes its argument of "rightindextipposition" to the leap.get object encapsulated within it, and thus the values collected from leap.get will be consistent with those specified in the leap.speed object. Through the use of argument passing, abstractions can be combined and expanded upon to develop more complex calculations and gesture measurements.

4.2 Gesture Taxonomy

A useful reference for the author during the development of the LeapMax library was the categorizations of gesture and posture described in Mitchell, Madgwick, and Heap's paper *Musical Interaction with Hand Posture and Orientation: A Toolbox of Gestural Control Mechanisms* [16]. Although this paper discusses these categorizations with relation to data returned by the Mi.Mu data glove system, the same concepts apply when using the Leap Motion. Especially valuable are the discussions on converting continuous data to discrete control mechanisms, and how continuous controls and discrete controls can be used in conjunction to develop "a state-based control system [and] to enable one-to-many gestural mappings" [16, pp. 2]. Abstractions associated with many of the gestural control mechanisms described by Mitchell and his coauthors are implemented in the LeapMax library including orientation control (leap.rotation), positional displacement (leap.difference, leap.stepper), and posture identification (leap.posture, leap.getposture).

Additionally, during the development of the LeapMax library and the LeapMax performance system, the author found it useful to consider measurable gestures in the context of several other categorizations which are described in the next sections. These categorizations are not intended to classify discrete gestures, nor suggest some inherent relationship between gestures measurements belonging to the same category. Rather, these categories represent multiple contexts in which gesture data can be measured, and assist in developing methodologies for structuring the LeapMax library.

4.2.1 Data Types

The types of data extrapolated by the LeapMax library can be divided into three main categories: position based data, movement based data, and acceleration based data. Although they are all interdependent, each data type is linked to gesture in a different way. Determining which type of data is the most symbolically linked to an output is vital in order to create meaningful relationships between the actions taken by the user and the output of the system.

- *Position*

  Position data (leap.get) is based on the spatial state of the system (ie. the hands) at a given point at time. This data is particularly effective in developing a

sense of space within the context of the global Leap frame.

- *Movement*

  Movement based data (leap.velocity, leap.grabspeed) is measured by taking the derivative of position data with respect to time. This type of data can be associated with the amount of energy in a gesture.

- *Force*

  Force based data (leap.acceleration) is measured by taking the derivative of velocity data with respect to time. Because force is required for acceleration, this type of data can be related to the amount of force being exerted in a gesture.

4.2.2 Coordinate Frames

The second categorization of gesture the author found useful while developing the Leap library was by reference to coordinate space. Largely, gesture data exists within one of three coordinate spaces: global coordinate space, local hand space, and relational space.

- *Global*

  Position data from leap.get and velocity data from leap.velocity both represent values which reference the global coordinate space. The global coordinate system is static with respect to the performer and thus any movement of the hand can be detected in this space.

- *Hand*

  Values extracted from leap.fingerbend and leap.grab represent data that exists in the hand space. This category is associated with the posture of a hand, and because it is calculated with respect to the hand's coordinate system, the actual position of the hand in space does not have an effect on these values.

- *Relational*

  Values computed by combining and comparing multiple data points, such as values extracted from leap.difference and leap.distance represent values measured in relational space. Relational values can also be computed from directional data collected from the leap (ie. finger direction, palm normal, etc) by using dot or cross products on multiple vectors. Generally, the values being compared in these computations are actually measured in either the global or hand coordinate systems, but by combining them during calculation a relational frame is established.

--

These three categories represent three contexts of gesture measurement which, while all interdependent, can also be manipulated and considered in isolation. For example, palm position, which is measured in the global frame, can be controlled independently from fingerbend, which is measured in the hand frame. The user can bend their finger without moving their palm and move their palm without changing the bend of their finger. Multi-mapping using values from different coordinate spaces is particularly effective because of the freedom of control that this independence allows.

4.2 Measuring Posture

The LeapMax's posture measurement system is predicated on the simple kinematic finger model created during the development of the Digits IR hand tracking system [5]. This model leverages the natural interdependencies between the three joints in each finger to infer and calculate the bend of each joint based on a single value. In LeapMax, the single value used to measure finger bend is the dot product calculation of the rotation vector of the innermost bone of each finger and the normal vector of the palm. Because this bone is perpendicular with the palm normal when the finger is fully extended and parallel with it when fully closed, this dot product calculation natively scales from 0 to 1. Although this value does not represent a full resolution description of the bend of the finger, it is accurate enough to allow for a wide range of measured hand postures.

To create a full hand posture measurement, the leap.posture object calculates a finger bend value for each finger on a hand and outputs those five values as an ordered list. The leap.getposture object accepts the posture list from the leap.posture object and allows a posture to be saved, and then compared to the current posture. The leap.getposture object outputs a value from 0 to 1 corresponding to how similar the current posture is to the saved one. Because the output of leap.getposture is a continuous value, it can not only be used for discrete posture selection, but also as a value associated with the similarity of the current posture to a saved one.

5. THE LEAPMAX PERFORMANCE SYSTEM

The LeapMax performance system is a project built using the LeapMax API and library which demonstrates an example hardware setup as well as a use case of the LeapMax software as a gestural DMI.

5.1 Hardware

The LeapMax performance system consists of a Leap Motion device mounted to a pair of non-prescription glasses using the Leap Motion VR mount. A BNO055 absolute orientation sensor is attached to the top of the Leap mount and is used to track the Leap's orientation in space. This sensor is interfaced with the computer using a Teensy LC microchip which sends orientation data as Euler values through serial. Although there are means of communicating orientation and other data wirelessly, because of the size and complexity of the data transmitted by the Leap device, a wireless system

Fig. 5

was not considered during the development of this project. Instead, two fifteen foot USB cables are used to extend the range of the headset. These wires are pinned to the shirt of the performer to reduce the weight of the device when worn.

Fig. 6

A major goal for the hardware design was a reduced cost for the system and this goal is reflected in the final implementation of the design. At the time of writing, the Leap Motion device with the mount costs just under $100, the BNO055 sensor costs $35, and the Teensy LC costs $15, putting the starting cost of this project at under $150. The minimal amount of peripheral technologies required for use also greatly improves the reliability and consistency of the hardware system as a whole.

5.2 Semiotic And Free Gesture Design

Claude Cadoz argues that instrumental gesture can be described both by its interaction with the physical environment and by its function in the communication of information [8]. He identifies three interdependent functions that describe this relationship: the ergotic function, the epistemic function, and the semiotic function. The ergotic function deals with "material action, modification and transformation of the environment" [8, pp. 78]. This function focuses solely on the forces applied within a performance environment by a gesture, such as, for example, striking or blowing an instrument. The epistemic function focuses on the feedback and the reaction that is received from the environment during a gesture (e.g. how hard was it to move an object, what texture was the object, etc). Finally, there is the semiotic function, which deals with the communicative intent of a gesture. According to Cadoz, the semiotic function is "the only function associated to gesture in the sense of free- or empty-handed gestures - sign-language, natural gesture, gesticulation, pantomime, etc" [8]

Most instruments and interfaces, by the nature of their physical manifestation and appearance, afford particular gestures for the performer. Buttons are meant to be pushed, strings can be plucked or bowed. Even more complex sensor systems such as the Mi.Mu data glove, although not immediately clear in their control methods, give some indication of potential interaction based on the physical device and placement of sensors.

Because LeapMax uses no physical interface or sensors, no obvious interaction method is immediately presented, and thus the relationship between gesture and the resulting output, be that sound, visuals, or otherwise, becomes even more vital as a method to relay to the user and the audience the potential methods of interaction with the system, especially if they are not already familiar with the it. This adds an extra layer of complexity when designing systems that are both rich and complex, but also intuitive and understandable.

5.3 Results, Reflections, and Future Work

Regarding the development of the Lady's Glove, Sonami remarks that in order to emphasize the sublime relationship between her gestures and the sound produced, she hid the wires within the glove in order to "make it look magical" [11]. The completely freehanded interaction of the LeapMax system also makes the user feel and look magical.

The design choices made while developing the LeapMax system were validated through the data collected using the performance system, which was found to be highly accurate and consistent. Setting the Leap to its head mounted mode is vital in order to retrieve good tracking data because this mode uses a different tracking algorithm than the desktop mode. In head mounted mode the Leap is much better at estimating the positions of occluded fingers, although the system still has trouble discerning when one hand occludes another.

One significant limitation of the LeapMax performance system is its ability to track hands only when they are placed within a 150 degree region in front of the Leap sensor which makes some full body gestures untrackable. This limitation can be reduced, however, by always keeping the Leap device (e.g. the head of the user) pointed toward the hands being tracked.

Future iterations of the LeapMax performance system could potentially include multiple Leap motion sensors working in unison to increase the detection range of the system. Additionally, developing a wireless solution would greatly improve the mobility of the device and also reduce the constriction that comes with wired systems.

In the future, the author hopes to incorporate real time machine learning technologies such as Wekinator into the LeapMax system to facilitate more precise and open ended gesture detection and recognition.

6. Conclusion

The LeapMax project can be broken down into three parts. Firstly, the LeapMax API is a Max C external which presents an updated and optimized protocol for communicating Leap Motion data to Max. This API is optimized through the use of Max dictionary data structures. The LeapMax library augments the data passed to Max by the API through a series of modular abstractions which extrapolate more complex gestures and postures. Finally, the LeapMax performance system demonstrates a use case of this technology as a gestural interface for a digital musical instrument.

The work presented in this paper represents the technical foundations for an increased exploration of the Leap Motion in conjunction with Max as a gestural interface. The hope of the author is that the LeapMax project will continue be used by other performers, developers, and researchers to further refine and explore the capabilities of the system.

7. References

[1] E. S. Silva, J. A. O. de Abreu, J. H. P. de Almeida, V. Teichrieb, and G. L. Ramalho, "A preliminary evaluation of the leap motion sensor as controller of new digital musical instruments," *Recife, Brasil*, 2013.

[2] "AirHarp – Leap Motion Gallery." [Online]. Available: https://gallery.leapmotion.com/airharp/. [Accessed: 25-Oct-2018].

[3] Akamatsu M., "aka.objects | akalogue." [Online]. Available:http://akamatsu.org/aka/max/objects/. [Accessed: 25-Oct-2018].

[4] S. D. Thorn, “Alto. Glove: New Techniques for Augmented Violin.” *(NIME)* [Online]. Available: http://www.nime.org/proceedings/2018/nime2018_paper0070.pdf
[5] D. Kim *et al.*, “Digits: freehand 3D interactions anywhere using a wrist-worn gloveless sensor,” in *Proceedings of the 25th annual ACM symposium on User interface software and technology*, 2012, pp. 167–176.
[6] “Flocking – Leap Motion Gallery.” [Online]. Available: https://gallery.leapmotion.com/flocking/. [Accessed: 25-Oct-2018].
[7] “Geco MIDI – Leap Motion Gallery,” *Leap Motion*. [Online]. Available: https://gallery.leapmotion.com/geco-midi/. [Accessed: 25-Oct-2018].
[8] C. Cadoz and M. M. Wanderley, *Gesture-music*. 2000.
[9] J. Ratcliffe, “Hand motion-controlled audio mixing interface,” *Proc. of New Interfaces for Musical Expression (NIME) 2014*, pp. 136–139, 2014.
[10] L. Hantrakul and K. Kaczmarek, “Implementations of the Leap Motion in sound synthesis, effects modulation and assistive performance tools.,” in *ICMC*, 2014.
[11] L. Sonami, “Instruments – Lady’s Glove.” [Online]. Available: http://sonami.net/ladys-glove/. [Accessed: 25-Oct-2018].
[12] “Leap Motion C API: LeapC Guide.” [Online]. Available: https://developer.leapmotion.com/documentation/v4/index.html. [Accessed: 25-Oct-2018].
[13] J. Francoise, “Leap Motion skeletal tracking in Max.” [Online]. Available: https://www.julesfrancoise.com/leapmotion. [Accessed: 25-Oct-2018].
[14] “LyraVR – Music Reimagined.” [Online]. Available: http://lyravr.com/. [Accessed: 25-Oct-2018].
[15] “Mi.Mu,” *MI.MU*. [Online]. Available: https://mimugloves.com. [Accessed: 25-Oct-2018].
[16] T. Mitchell, S. Madgwick, and I. Heap, “Musical Interaction with Hand Posture and Orientation: A Toolbox of Gestural Control Mechanisms,” p. 5.
[17] “Power Glove,” *Wikipedia*. 08-Sep-2018.
[18] T. Place and T. Lossius, “Jamoma: A Modular Standard For Structuring Patches In Max” p. 4.
[19] https://developer.leapmotion.com/documentation/v4/concepts.html
[20] http://dsky9.com/rift/vr-tech-6dof/

8. Appendix

A github repository of the LeapMax project can be found here: https://github.com/cooperjones23/leapmax.git

A live performance using the LeapMax performance system can be viewed here: https://www.youtube.com/watch?v=ySnFUogCr8w&feature=youtu.be