Quantifying Touch: Examining the Differences Between Hardware and Software Input Methods in a Musical Context

Remy Pham

***Abstract*— There is a divide among some musicians today; there are those who believe that digital audio solutions are the optimal way to manipulate and interact with audio and music, and those who believe that their analog predecessors are instead ideal (i.e. a voltage-controlled hardware synthesizer). Modern 'point-and-click' music producers may justify their means through computational power and adaptability through a GUI, while old-fashioned hardware users may rationalize their preference with more subjective aspects, such as the value of tactile features and better accessibility of those features. This paper describes a study performed in which the two lines of thought are compared in a more objective light - through amplitude and frequency accuracy tests. A 40-user test was devised to examine behavioral differences. Performing the analysis showed that there were no significant relationships between input method and user test accuracy; however, there were several positive correlations between other behavioral traits of the users (such as trial timing and frequency of tone playback) and relevant permutations of the user error.**

I. BACKGROUND

A. Overview

Though the subjective nature of art is in some ways the antithesis of the objective nature of science, one may find they have intertwined and laid mutual influence on each other for as long as they have existed. The role of science and technology in the realm of music is no exception to this mutual influence. When observing the technological advances made over the years, analogous advances are seen to be made in chronological tandem in music (in the form of genres, sound design, etc.).

The modern musician's production environment, much like the environment of many other vocations, has evolved drastically as a result of these technological changes.

There exists a divide among many musicians today resulting from the present-day ubiquity of digital synthesizers and thus the decreased reliance and presence of their hardware predecessors. One side of this divide embraces the hardware tendencies of the past, citing the benefits of physical controls and tactility. Meanwhile, the other party claims that digital emulation of audio-related tasks is more than sufficient to replace our hardware dependencies of the past.

This section will briefly recapitulate the origins of audio synthesis and the genesis of this divide.

B. The Analog Origins of Synthesized Sound

The first instance of synthesizers that were designed for commercial use came in the 1970's as an indirect result of Moore's law. Roberg Moog, known as the father of the synthesizer, invented the first voltage-controlled synthesizer, which relieved the synthesizer of its motor-powered

Fig. 1. The Minimoog Model D Synthesizer

dependency. Moog's vision to modularize and abstract the complicated internals of a synthesizer was realized when he designed the Minimoog Model D Synthesizer (Figure 1), which later experienced great success in the commercial market, selling more than 13,000 units over the decade following its release. The influence of the Minimoog Model D and the ensuing competitors over this time could be seen in music studios around the world, and heard in countless tracks from the music of the 1980's.

At this point, hardware, with all of its tactile input features, dominated the music production process. As a result, the fashion in which the musician interacted with their audio manipulation process was largely tactile. Features of electronics at the time were adopted by these commercial hardware synthesizers; the likes of potentiometers, rotary encoders, and linear faders made up the face of the Minimoog and others.

But as computing capabilities continued to improve at scale, it would only be a matter of time until digital emulation would make its way into the industry.

C. The Digital Age

The limitations of hardware synthesizers became increasingly noticeable as the frequency of their use increased. Namely, the computational requirements for polyphony, or the ability to play multiple notes at once, posed an issue; generally speaking, playing multiple synthesized notes would require an instance of a hardware synthesizer for every additional voicing that the musician may desire.

Increasing computing power meant having the ability to emulate multiple instances of audio synthesis, a much more practical alternative to having multiple hardware instances. Furthermore, with the advent of the personal computer, a palette was provided for a graphical user interface which could be retrofit to represent hardware controls.

The eventual ubiquity of the personal computer and its increased capability for computation has resulted in a disruptive change in the way the modern musician interacts with audio manipulation. The ability to emulate a multitude of hardware synthesizers with software has made it such that hardware synthesizers now play a relatively niche role in the music production process.

D. The Persistence of Hardware

The advent of the Musical Instrument Digital Interface (MIDI) standard, which allowed physical controls to be mapped to software parameters, led to a resurgence in hardware use among musicians. A category of hardware known as the MIDI controller emerged; its function in practice provided the same relative experience in audio manipulation as a hardware controller, in that the MIDI controller's inputs would be mapped to parameters in the software application, typically functioning as a mixer or synthesizer.

E. The Growing Divide

The MIDI controller category of music hardware was in part born out of subset of musicians who believe that the analog predecessors are in many ways superior to today's software synthesis incarnations. Gaute Barlindhaug writes in his paper "Analog Sound in the Age of Digital Tools: The Story of the Failure of Digital Technology" that the blind replacement of all things analog as a software entity is not necessarily for the better, and that it is not necessarily true that "new is better." He cites the inability to easily access a parameter in a software GUI as a disadvantage, claiming that the more accessible hardware features of a hardware synthesizer are of greater benefit to the user. Additionally, even the imperfections of hardware synthesizers (such as phasing from imperfect hardware parts) are deemed an advantage by many, providing 'warmth' in sound that otherwise is inferiorly replicated in software (if even attempted). In short, Barlindhaug speaks on behalf of many when writing that the "first problem with musical software is the lack of tactile aspects."

II. INTRODUCTION

In its current state, the tools and procedures available to the modern musician are predominantly digital in nature. The accessibility of digital sound synthesis, among other functions, has made it possible to emulate hardware whose functions were otherwise once exclusively an interactively tactile matter.

The role of the GUI in modern music production practices is in turn vital to the process itself. Transitioning smoothly from a strictly physical interface to a digital GUI makes it imperative to bring the most important of the distinctive tactile features of hardware to the graphical user interface.

Fig. 2. The Sylenth1 Software Synthesizer

The most notable features of musical hardware with regard to user-facing tactile controls include potentiometers, rotary encoders, and hardware faders. The design of most current incarnations of software synthesizers and their GUI's draws from their storied past; most of the functional aspects of their design include graphical representations of the same types of controls. Generally one will find sets of vertical faders and banks of knobs which are each mapped to relevant parameters in the synthesizer. An example, the Sylenth1 software synthesizer, is shown in Figure 2.

III. MOTIVATION

The motivation for this project stems from the hardware-versus-software divide among a subset of modern musicians. When it comes to the arguments behind the rationales of either side, whether it be that of a hardware-wielding musician or a 'point-and-click' producer, one will find that they are generally subjective. As such, the nature of subjective advantages and disadvantages of hardware makes it difficult to come to a concensus. The goal of this study is to observe and examine the behavior of both hardware users and software users in a more objective light - through hard data.

A. Finding Relevant Quantifiable Metrics

In order to choose contextually appropriate data, we examined the environment in which the present-day musician may interact with either a hardware device or its software counterpart, as well as the functions with which they may be tasked.

When considering the modern musician's production environment, these tasks often involve the adjustment of values that control sound-related properties including, but not limited to amplitude, frequency, track panning, etc.

IV. METHODOLOGY

A. Overview of Test

The participants of the study were tasked with two tests: the amplitude test and the frequency test. The participant is presented with a GUI containing three buttons and a GUI vertical fader. Additionally, they were given either a MIDI

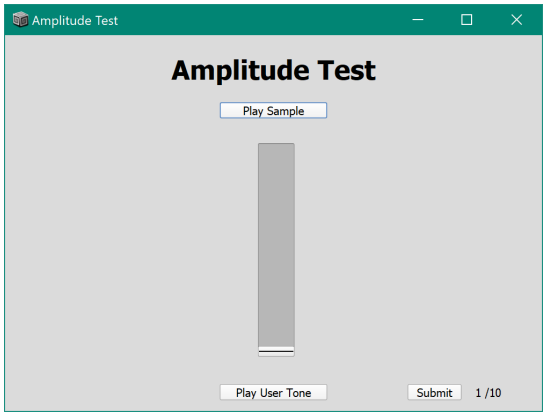

Fig. 3. The Amplitude Test GUI

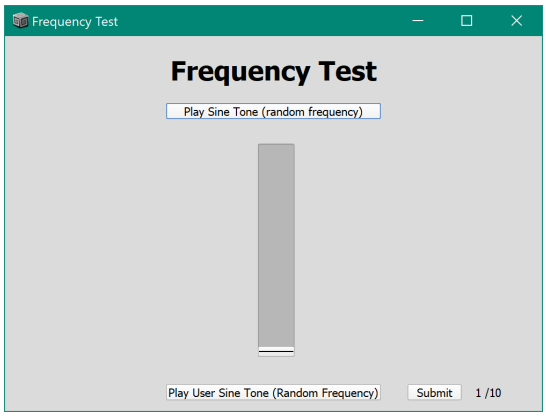

Fig. 4. The Frequency Test GUI

controller or a mouse (depending on the randomly assigned input method group) to interact with the GUI, and Audio Technica M50x headphones.

B. GUI Design

1) Amplitude Test Design: The goal for the participant in the amplitude test is to determine the volume of a sample as accurately as possible. Figure 4 shows the GUI of the Amplitude Test. The 'Play Sample' button allows for the user to play back a sample at a predefined and unknown volume (between -48 and 0 decibels). Pressing the 'Play User Tone' button plays back the same sample as the previous button with the exception that its volume is instead set at the fader's volume level. Finally, the 'Submit' button submits the user's current amplitude set by the fader as their final answer, and loads the next sample.

The task for the participant is to match the volume of the sound played with the 'Play User Sample' button with the sample playback from the 'Play Sample' button; in other words, the ideal fader setting would result in both of the playback samples being, and thus sounding the same. The user undergoes ten trials in this test; with each trial, both the sample and the volume of the sample are changed.

2) Frequency Test Design: The frequency test (Figure 5) works in a nearly identical fashion with two exceptions. Firstly, the playback buttons 'Play Sine Tone' and 'Play User Sine Tone' instead play a short (0.5 seconds) instance of a sine wave at a random frequency from 400Hz to 10kHz. Secondly, the vertical fader controls the pitch of the sine tone triggered with the 'Play User Sine Tone' button. The goal then for the frequency test is for the user to match the pitches of both playback buttons, similarly as before. The user again goes through ten trials, each with a different and random sine tone set at a random pitch.

Though the output of the faders begins as a choice on a linear scale between the values 0 and 1, it is translated according to the test in which it belongs. The justifications for these choices is described later in the section 'Considerations'.

3) Implementation: The GUI and synthesis functionality of the test was written in SuperCollider, a language that abstracts the process of generating instances of synthesis. It does so by having a local server available such that the language itself acts as a series of function calls; doing so allows for the ability to create instances of synthesized waves such as sine oscillators with a single line of code, without having to write the math itself. SuperCollider also provided the framework for the GUI.

The choice for the fader as a common grounds for interaction with the test was made because of its historical significance as a primary means of tactile audio manipulation, as evinced by its graphical reproduction in most modern software synthesizers. Additional characteristics and limitations of this study's implementation of the GUI are further discussed in the 'Considerations' section.

V. Data

A. Control Groups

There were 40 users who participated in the study. Each member was randomly assigned a means of input - the two options include either using the mouse as input to control the GUI, or using an Akai MIDImix MIDI controller as input. This made for two groups, mouse users and MIDI users, consisting of twenty users each.

B. Amplitude Test Data

The primary metric of the study is the accuracy of the participant's submitted fader values versus the playback sample's true amplitude. In the amplitude test, this primary metric is referred to as the error, which is defined as the difference between the user's submitted amplitude and the true amplitude in decibels.

While this metric alone lays the foundation for the study's motivation, the implementation of the test in SuperCollider made for an environment conducive to collecting more data with regards to the participant's behavioral tendencies. As such, the following metrics were also recorded for each trial of the amplitude test: the number of button presses for the 'Play Sample' button, the number of button presses for the 'Play User Sample' button, and the duration of each trial in seconds.

As the decibel metric is the perceptive metric of volume, we also considered the user's errors in terms of amplitude as well. This relationship is further discussed in 'Considerations'.

C. Frequency Test Data

The frequency test follows the same design and procedural identity as the amplitude test. Thus, the button press count and timing metrics were collected identically as they were in the amplitude test.

The error was also similarly collected, but in the frequency test, the unit is Hertz, used to define the frequency of the sine wave instance. However, the relationship between frequency of a waveform and perceived pitch provides reason to also consider the square root of this error for this particular test, among other permutations of the error data. This relationship is further discussed in the 'Considerations' section (this is analogously similar to the amplitude test's decibel and amplitude metrics).

D. Subjective Data

In addition to the aforementioned metrics collected by the test, the users were also prompted to describe their subjective level of musical expertise on a scale of one to ten. In particular, the participants were prompted with the following question: "On a scale of one to ten, one being you are completely inexperienced with music from an experiential or observational standpoint, and ten being you are extremely experienced with music, what number would you choose?" This metric was added in order to attempt to account for possible discrepancies in user behavior and accuracy due to experience.

E. Relationships Among Data

With all of the data that the test collects from the participants, we considered several relationships between the variables. Firstly, we consider if there are any statistically significant differences between the accuracy of participants using the mouse input versus the participants using the MIDI controller in both the amplitude tests and the frequency tests. Their subjective musical experience score was also included as a factor.

The other behavioral data collected, namely the number of button presses for each individual playback button and the length of each trial for both tests was analyzed in order to determine if they exist as a function of the method of input as well as the user's subjective music experience score.

VI. ANALYSIS

A. Amplitude Test

In the amplitude test, the user is the subject of the analysis, and the repeated variable is the test trial (1-10). It follows the Scaled Identity Covariance structure.

1) Dependent Variables: The dependent variables in the amplitude test to be tested include the user error in decibels *error*, the absolute value of the error *abserror*, the error converted to amplitude *errorAmplitude*, and the difference of amplitudes (user's guess - true value) $errorDiffAmplitude$. The conversion from decibel to amplitude used is $amp = 10^{((dB/2)/10)}$.

2) Factors: The factors (whose functional role is the qualitative variable) were the participant's input method (MIDI controller or mouse) *mouseOrMIDI*.

3) Covariates: The covariates include the frequency of each button press in the amplitude test $numClicksUserSample$ and *numClicksRandomSample*, as well as the timing of each trial $timing$ and the user's subjective musical experience score $subjectiveScore$.

B. Frequency Test

In the frequency test, the user is again the subject of the analysis, with the test trial being a repeated variable. It also follows the Scaled Identity Covariance structure.

1) Dependent Variables: The first dependent variable analyzed was the *error* of the user, defined as the difference between the user's attempt (in Hz) and the sine tone's true frequency (per trial). Then, as justified in the section 'Considerations', the absolute value of the error *abserror* and the square root of the absolute value of the error *sqrterror* were analyzed as dependent variables, as well as the difference of the squares $diffsqrterror$.

2) Factors: The factors were the participant's user input method (either MIDI controller or mouse) *mouseOrMIDI*.

3) Covariates: The quantitative variables in the study included the the number of clicks registered by the user with regards to playing the reference sine tone *numClicksRandomTone* and their own user-defined sine tone $numClicksUserTone$. The $timing$ was also included, as well as the user's subjective musical experience score $subjectiveScore$.

VII. RESULTS

A. Amplitude Test

1) Error as the Dependent Variable: The first analysis performed with the amplitude test data uses the previously mentioned variable definitions. The dependent variable is set to *error*. The following p values are found:

$$p(mouseOrMIDI) = .842$$

$$p(subjectiveScore) = .468$$

$$p(numClicksUserSample) = .128$$

$$p(numClicksRandomSample) = .092$$

$$p(timing) = .878$$

With a p value of .05 indicating a significant relationship between the data, we find that in this first test, the quantitative and qualitative variables do not bear a significant relationship with the user's *error*.

2) Absolute Value of Error as the Dependent Variable: Next, the absolute value of the error (in dB) was analyzed as the dependent variable.

$$p(mouseOrMIDI) = .366$$

$$p(subjectiveScore) = .197$$

$$p(numClicksUserSample) = .930$$

$$p(numClicksRandomSample) = .537$$

$$p(timing) = .001$$

While the results are largely the same with regards to significant relationships between the factors and covariates with *abserror*, we do see that there is a positive correlation between the timing of the trial and *abserror*. In other words, users who spent more time on the trial had a tendency to have a closer guess as to the true volume of the sample.

3) Error Amplitude as the Dependent Variable: Using instead the absolute value of the error's dB value converted to an amplitude, we have the following p values:

$$p(mouseOrMIDI) = .446$$

$$p(subjectiveScore) = .107$$

$$p(numClicksUserSample) = .979$$

$$p(numClicksRandomSample) = .462$$

$$p(timing) = .002$$

As we saw with the second test, the timing of the trial again bears a significant relationship with the true amplitude of the sample, while the other factors and covariates do not.

4) Difference of Amplitudes as the Dependent Variable: Using the difference of the user's sample dB and the true dB, each converted to their linear amplitude, the following p values are found:

$$p(mouseOrMIDI) = .924$$

$$p(subjectiveScore) = .796$$

$$p(numClicksUserSample) = .170$$

$$p(numClicksRandomSample) = .111$$

$$p(timing) = .807$$

Taking the difference of true amplitudes did not bear a significant relationship with the factors and covariates.

B. Frequency Test

1) Error as the Dependent Variable: The first test is performed with the aforementioned variable definitions with the error as the dependent variable. The p values yielded for *subjectiveScore*, *numClicksRandomTone*, *numClicksUserTone* and *mouseOrMIDI* are as follows:

$$p(mouseOrMIDI) = .190$$

$$p(numClicksUserTone) = .659$$

$$p(numClicksRandomTone) = .759$$

$$p(subjectiveScore) = .660$$

$$p(timing) = .255$$

Using .05 as our p value again, we find that there are no statistically significant correlations between the frequency of the user's button presses and/or their input method identification versus the user error.

2) Absolute Value of Error as the Dependent Variable: Next, we analyzed the absolute value of the error (in Hz). This choice was made with similar reasoning as in the amplitude test analysis, in that here we concern ourselves only with the magnitude of the perceptive error, disregarding the sign.

$$p(mouseOrMIDI) = .270$$

$$p(numClicksUserTone) = .743$$

$$p(numClicksRandomTone) = .672$$

$$p(subjectiveScore) = .712$$

$$p(timing) = .534$$

This test bore similar results as the first test.

3) The Difference of Squares as the Dependent Variable: The third test replaces error with the difference of the square root of the user's frequency guess and the true sine tone frequency respectively. Running the analysis shows relatively similar results:

$$p(mouseOrMIDI) = .175$$

$$p(numClicksUserTone) = .755$$

$$p(numClicksRandomTone) = .691$$

$$p(subjectiveScore) = .658$$

$$p(timing) = .377$$

Again we find that there are no significant correlations. It is worth noting that this change in dependent variable assignment led to a slight increase in correlation in some of the factors and covariates compared to the first test.

4) The Square Root of the Error: The fourth test uses the square root of the absolute value of the error $sqrterror$. We find the following p values:

$$p(mouseOrMIDI) = .351$$

$$p(numClicksUserTone) = .129$$

$$p(numClicksRandomTone) = .036$$

$$p(subjectiveScore) = .992$$

$$p(timing) = .311$$

Here we can see that taking the square root of the absolute value of the user error bears a statistically significant correlation with the frequency of the user's custom sine tone playback presses *numClicksRandomTone*.

VIII. CONCLUSION

In conclusion, we have found that the input method did not bear a significant effect on the user's error, as well as any relevant permutations of their error, despite the discrepancy in granularity between the two methods. There were two metrics that did bear a significant relationship with the user error, one in each test.

In the amplitude test, we found that the time spent on each trial correlated with the absolute value of the user error in both decibels and true amplitude. Specifically, the longer the user spent on a given trial, the more likely they were to minimize the absolute value of their error in decibels and amplitude.

In the frequency test, it was discovered that there is a significant correlation between the number of times the user listens to the reference sine tone and the square root of the absolute value of the error.

One possible explanation for the discrepancy in factor/covariate correlation consistency between the two tests is the nature of volume-sensitivity and pitch-sensitivity. Spending more time on a trial led to a higher likeliness of lesser error in the amplitude test, while we are not able to say the same for the frequency test. In other words, spending more time on a trial was more conducive to volume-sensitive tasks, while it did not improve the user's pitch-sensitivity. Alternatively, it shows that higher interaction with the playback ability in the frequency test (particularly the ability to play back the reference tone) was more conducive to minimizing the square root of the absolute value of the error.

While the input method classification did not bear any significant correlations with any permutations of user error, along the way we were able to discover some interesting relationships between other behavioral tendencies of the user and the error's permutations.

IX. CONSIDERATIONS

A. Implementations of GUI Faders

It is worth noting that the fashion in which the point-and-click user interacts with the features of a software synthesizer can be drastically different than its hardware predecessor, despite similarities in visual design. For example, a physical potentiometer or rotary encoder would naturally be rotated by hand in a circular fashion; however, this is almost always not the case when interacting with a GUI knob. In fact, interaction with a GUI knob is almost exactly similar to that of a GUI vertical slider,. A point-and-click user would need to first initialize the interaction with the GUI knob by clicking down on the feature, and perform the rotation by dragging the cursor up vertically. The y-axis movement would then be translated to a circular rotation.

B. Discrepancies in Granularity of Input Method

One of the main differences between a hardware MIDI control versus its graphical counterpart is the difference in granularity. Each fader or potentiometer whose signal is received through the MIDI protocol is translated to a whole number value between 0 and 128. A software synthesizer GUI would typically provide a much finer granularity with respect to the parameter the graphical fader or knob controls. This is due to factors such as pixel density and the size of the virtual slider. There are some software synthesizers that allow for text input to be entered as a value for the desired parameter. Additionally, further granularity can be reached through a graphical fader in particular by extending the relationship between the cursor and the fader graphics beyond a 1:1 scale; this is impossible on hardware controls due to physical limitations.

C. Amplitude and Frequency Test Fader Range

Human perception of audio-related characteristics such as amplitude and frequency differs from the nature of the characteristics themselves. In short, a linear change in either of these facets does not necessarily result in a perceived linear change by the listener. It is for this reason that the fader in the GUI does not linearly increase the amplitude and frequency levels of each respective test. Instead, the amplitude fader's value (from 0-1 linearly) is translated into the appropriate decibel value; similarly, the frequency test's fader value raises exponentially from within the range of 400Hz to 10kHz. Considering the feedback-loop nature of the test, both of these choices were made in order to allow for the user to adjust the appropriate parameter in a perceivably linear fashion. Furthermore, this translation from a linear value into a value appropriate for the domain at hand is common in modern audio synthesis software. The range for the frequency fader was chosen for its overlap with the frequencies most easily perceived by humans; sub-bass frequencies were omitted, as well as frequencies over 10kHz as the two bounds approach the limits of human hearing.

X. ACKNOWLEDGEMENT

I would like to thank David Kant of the UCSC Music Department for his direction in creating the testing protocol for this study. I would also like to thank Professor Douglas Bonnett for his guidance in the data analysis portion of this project. Finally, I would like to thank Patrick Mantey, the advisor to this Master's Project, for his advice and instruction regarding the development of the study.

REFERENCES

[1] Barlindhaug, Gaute. "Analog Sound in the Age of Digital Tools: The Story of the Failure of Digital Technology".

[2] Jensenius, Alexander Refsum, and Michael J. Lyons. A NIME Reader: Fifteen Years of New Interfaces for Musical Expression. Springer, 2017.

[3] MusicRadar. (2018). Why should computer musicians use hardware?. [online] Available at: https://www.musicradar.com/news/tech/why-should-computer-musicians-use-hardware-542544 [Accessed 27 Aug. 2018].

[4] Krzysztof Marasek, Andrzej Romanowski and Marcin Sikorski. Emerging Trents and Novel APproaches in Interaction Design. Proceedings of the Federated Conference on Computer Science and Information Systems, 2017.