Syntacts: Open-Source Software and Hardware for Audio-Controlled Haptics

Evan Pezent, *Student Member, IEEE,* Brandon Cambio, *Student Member, IEEE,* and Marcia K. O'Malley, *Fellow, IEEE*

***Abstract*—As vibrotactile feedback systems become increasingly commonplace, their application scenarios are becoming more complex. We present a method of vibrotactor control that accommodates emerging design requirements, namely large vibrotactor arrays that are capable of displaying complex waveforms. Our approach is based on control through digital audio interfaces. We describe a new open-source software and hardware package, Syntacts, that lowers the technical barrier to rendering vibrations with audio. We also present a tutorial on common control schemes with a discussion of their advantages and shortcomings. Our software is purpose-built to control arrays of vibrotactors with extremely low latency. In addition, Syntacts includes means to synthesize and sequence cues, and spatialize them on tactile arrays. The Syntacts Amplifier integrates with the audio hardware to provide high-quality analog signals to the tactors without adding excess noise to the system. Finally, we present results from a benchmarking study with Syntacts compared to commercially available systems.**

***Index Terms*—vibrotactor, audio, control, open-source**

I. Background

One of the most important and ubiquitous modes of haptic feedback is vibration. Haptic vibrations are commonly delivered to users through small actuators known as vibrotactors, or simply tactors. Vibrotactors come in many forms, such as eccentric rotating mass (ERM) actuators, linear resonant actuators (LRA), voice coil actuators, and Piezo actuators. For several decades, vibrotactile feedback has been used extensively across a wide variety of applications, most notably mobile and wearable devices [1].

The modern era of vibrotactile research is faced with a number of new needs and requirements. For instance, the field has recently begun moving away from providing users with simple alert type cues to delivering salient cues rich in information. Many researchers are now designing devices with larger numbers of tactors integrated into single interfaces such as bracelets, armbands, and sleeves [2]–[4], full body suits and clothing [5], [6], and chairs [7]. Unfortunately, driving many vibrotactors simultaneously has traditionally been a difficult task for engineers and non-engineers alike due to the technical skills required, interfacing difficulty, or cost of equipment. Further, high-density arrays require more sophisticated rendering algorithms. *Spatialization*, or the manipulation of several actuators in an array-based on the placement of a virtual target location, has been explored to some extent [7].

The authors are with the Mechatronics and Haptic Interfaces Lab, Department of Mechanical Engineering, Rice University, Houston, TX 77005, USA. Emails: {epezent, btc6, omalleym}@rice.edu.

Manuscript received January 15, 2020; revised May 9, 2020.

In addition to increasing actuator counts, some vibrotactile research has recently focused on delivering complex vibration waveforms, beyond simple buzzes, to convey more meaningful information to users [8], or to more accurately simulate real-world phenomena [9]. The synthesis of such cues, however, is not a trivial task, with some researchers resorting to pre-recorded libraries [10] or high-level creation tools [11], [12]. Finally, while the advent of mainstream virtual reality (VR) systems has introduced new opportunities for vibrotactile feedback, it has also imposed additional constraints on control including low latency [13] and the need to alter cues on the fly in response to virtual events.

This paper aims to highlight a method of vibrotactor control that accommodates many of the these requirements and deserves detailed attention: *control through digital audio interfaces*. We present a new open-source software and hardware package, **Syntacts**, that lowers the technical barrier to synthesizing and rendering vibrations with audio. In Section II, we discuss common vibrotactor control schemes along with their advantages and shortcomings. Section III provides an overview of the hardware requirements for audio-based control, underscoring some of the lesser known details that can have a high impact on control, and introduces the Syntacts Amplifier board. In Section IV, we discuss software for audio-based control and then present the Syntacts software library. Finally, in Section V, we provide comparisons between Syntacts-based audio control and other methods. Conclusions and areas for future work follow in Section VI. Syntacts software and hardware designs are freely available at: www.syntacts.org.

II. Introduction to Vibrotactor Control

Because vibrotactors have been a staple of haptics for a long time, there exist many scenarios and approaches for their control. A typical *research-oriented* scenario requires controlling vibrotactors from a PC that may also coordinate an experiment, record data, and/or render visuals. Within this context, we summarize a few possible control strategies.

A. Function Generators

The simplest control implementation uses a standalone function generator connected directly to the tactor. This is easy because generators are purpose-built to output oscillating signals and envelopes, and can often meet the tactor's power requirements. However, function generators are limited in cue design, output channel count, and may be challenging to integrate with custom software. For these reasons, they are a poor choice for complex control.

B. Integrated Circuits

To serve the mobile device market, specialized integrated circuits (IC) have been developed for vibrotactor control. These ICs often handle both signal generation and power amplification, making them an all-in-one package. A common chip, the DRV2605L from Texas Instruments (TI), features a built-in library of effects that can be triggered and sequenced through $\mathrm{I^2C}$ commands. Some ICs are capable of closed-loop control which automatically detects the tactor's resonant frequency and can provide faster response times. The utility of ICs for laboratory research, however, is restricted by the need to design and fabricate custom PCBs, since their small package sizes make it difficult to prototype on breadboards (though preassembled PCBs and breakouts can be found in various online shops). Controlling many tactors becomes complicated and usually requires additional components such as multiplexers. Finally, PCs generally do not provide an $\mathrm{I^2C}$ interface, so a USB adapter or microcontroller (e.g., an Arduino) must be introduced to broker communication between the PC and ICs.

C. Dedicated Controllers

Unlike other actuators such as DC motors, there exist very few off-the-shelf, plug-and-play controllers for vibrotactors. One product marketed as such is the Universal Controller from Engineering Acoustics, Inc (EAI). It is designed to drive their ubiquitous C2 and C3 voice coil actuators, but can drive other tactors with similar load impedance. The controller interfaces to a PC via USB and can output up to eight individual channels, though the datasheet and our own testing (Section V) indicates that only four can be driven simultaneously. EAI provides a GUI and C API with adequate cue synthesization features, so integrating the controller with custom software is straightforward. The major downside of this controller is a very high upfront cost (approximately $\$2,250$) that not all researchers are willing or able to afford.

Texas Instruments also sells the DRV2605LEVM-MD, an evaluation module for the DRV2605L mentioned above, that could be considered a controller unit. The board integrates eight DRV2605L ICs, an I2C multiplexer, and a USB interface. Unlike the EAI controller, no high-level communication API is available, so either low-level serial programming or $\mathrm{I^2C}$ brokerage is still required to integrate it with a PC. Finally, a recent startup, *Actronika*. aims to sell a haptic processing unit, the Tactronik; however, details are currently sparse.

D. Audio Output Devices

Another approach to driving tactors, and main focal point of this paper, is through *digital audio output devices*. This approach hinges on the understanding that some vibrotactors, particularly LRA and voice coil variants, operate very similarly to headphones or loudspeakers. Like speakers, these tactors consist of an electrical coil within a magnetic field. Energizing the coil induces a magnetic force that, in the case of speakers, drives a cone to generate sound pressure, or, in the case of vibrotactors, drives a mass to generate vibrations. As such, the same hardware that drives loudspeakers can also drive vibrotactors with a few adjustments and considerations.

The technique of using audio to drive haptic actuators is simple yet relatively underutilized within the field. Outside of a few workshops [14], [15], the process has received limited documentation or comparison with existing control solutions. The remainder of this paper will discuss the implementation of audio-based control while introducing a new open-source hardware and software solution, **Syntacts**. We will show that using this approach can provide a number of benefits including relatively low implementation cost, support for large channel counts, and ultra-low latency.

III. Hardware for Audio-Based Control

A. Sound Cards / Digital-to-Analog Converters

The most important piece of hardware for audio-based control is the digital-to-analog converter (DAC) device. The DAC is responsible for converting digitally represented waveforms, like music files, to analog signals to be played though headphones or speakers. Virtually all PCs have a DAC integrated into the motherboard that outputs two analog signals through a headphone or line out jack (typically a 3.5mm phone jack) for left and right audio channels. If no more than two vibrotactors are needed, use of the built-in headphone jack may be sufficient for some users.

Driving more than two channels generally requires a dedicated DAC, or sound card. The least expensive options are *consumer grade* surround sound cards, which can be had in typical PCI-e or USB interfaces. Up to six tactors can be driven with 5.1 surround sound cards, while up to eight can be driven with 7.1 surround sound cards. We have found this to be a viable solution if consideration is given to differences between channel types (e.g., subwoofer channels are usually tuned for lower impedance loads than speaker channels). Offerings from Creative Soundblaster and Asus are among the most readily available choices. There also exist *professional grade* audio interfaces with more than eight outputs, such as the MOTU UltraLite-mk4 and 16A with 12 and 16 channels, respectively. For even higher channel counts, the purely analog output MOTU 24Ao is a popular choice [16], [17]. A single unit provides 24 output channels, and up to five units can be connected using Audio Video Bridging (AVB) to drive 120 vibrotactors if desired. It should be noted that some professional devices may feature other I/O channels (e.g., MIDI, S/PDIF, etc.) that are of little use for driving tactors.

An *extremely* important consideration in sound card selection is the device's driver API support. An API describes a digital audio transmission protocol, and most drivers support many different APIs. Windows standardizes at least four first-party APIs: WDM-KS, WASAPI, MME, and DirectSound. As shown in Fig. 1, not all APIs are created equally. Because MME, which exhibits highly perceptible latency, is usually the default API, it could be easy to conclude that audio is insufficient for realtime haptics. Steinberg's third-party ASIO driver is widely considered to be the most performant option, but it is often only implemented by professional grade equipment. Regardless, API selection is a rather opaque setting under Windows, and appropriate software is usually required to select the preferred driver API (see Section IV). Driver

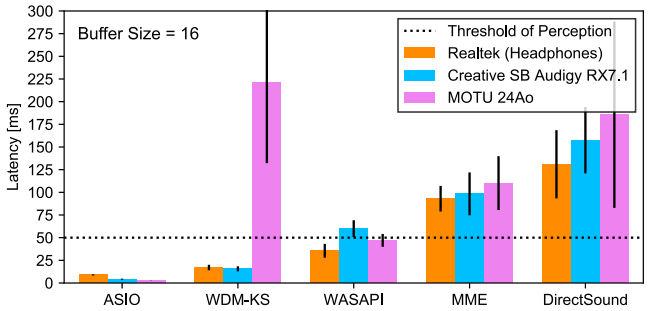

Figure 1: Mean Windows audio driver API latencies with standard deviation. Data collection methods are described in Sec. V. For reference, the dashed line indicates the perceptional threshold of visual-haptic simultaneity [13].

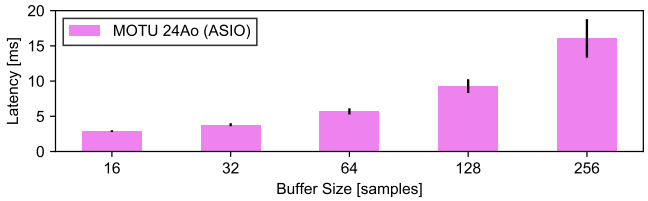

Figure 2: The effect on latency due to changing audio buffer sizes.

API selection is less of an issue on macOS, with CoreAudio being the universally recommended option. Another important consideration is audio buffer-size, or the number of audio samples sent on every transmission to the device. If the host PC has sufficient processing speed, smaller buffer sizes should be preferred for low latency (Fig. 2).

B. Amplifiers

Audio DACs typically output a low-power signal at what is called "line-level" because they expect that the output device will amplify the signal before it is actually played. Vibrotactors are similar to typical 8 to 16 Ω speakers, and therefore require amplification. Amplifiers are divided into different classes based on how they operate. *Digital* Class D amplifiers are the most common. They expect an analog input signal and output an amplified version of the signal with pulse-width modulation (PWM). This type of amplification tends to be very power efficient, but high-frequency PWM switching can add large amounts of electrical noise to a system. This is especially true when designing for arrays of vibrotactors, where multiple naively implemented Class D amplifiers can create enough noise to be physically felt. Class A, B, and AB amplifiers are *linear* amplifiers. These amplifiers tend to have much lower efficiency than the Class D, which can lead to heat problems if their thermal design is overlooked. However, because they do not constantly switch at high frequencies, they introduce considerably less noise into the overall system. Finally, a stable power supply is critical to the amplifier's ability to condition the signal. Batteries or linear power supplies provide much more stable power than typical switch-mode power supplies and allow amplifiers to operate with less noise.

Noisy power amplification can have detrimental effects on the performance of haptic devices that integrate sensors. For example, the first iteration of Tasbi's [3] tactor control hardware featured three commercial stereo Class D amplifiers powered by a generic switch-mode power supply. The high-frequency content emitted by these components resulted in errant motor encoder readings and noisy analog force sensor measurements beyond usability. As another example, we have noticed considerable noise emission from the C2 tactors and EAI Universal Controller (which also uses switching amplifiers) in MISSIVE [18] during EEG measurements.

C. Syntacts Amplifier

Based on these difficulties and limited commercial options for high-density output, we designed the purpose-built, eight channel **Syntacts Amplifier** board (Fig. 3). It is based on the TI TPA6211A1-Q1 3.1W audio power amplifier IC, featuring a Class AB architecture and fully differential inputs and outputs that together eliminate all noise issues we have experienced with commercial options. The Syntacts amplifier can drive small to medium sized vibrotactors with load impedances above 3 Ω from a 5V power supply at typical vibrotactile frequencies, making it suitable for many applications (Fig. 4). We have successfully tested it with various LRAs, EAI's C2 and C3 voice coil actuators, and Nanoport's TacHammer actuators. The amplifier is not intended for use with ERM actuators, which are trivially powered with DC voltage, nor Piezo actuators, which require higher voltages or custom controllers altogether. The Syntacts amplifier has thermal and short circuit protection and operates at voltage levels generally considered safe. However, potential users should understand that it has not undergone the testing required of commercial devices, and should take this into their safety considerations.

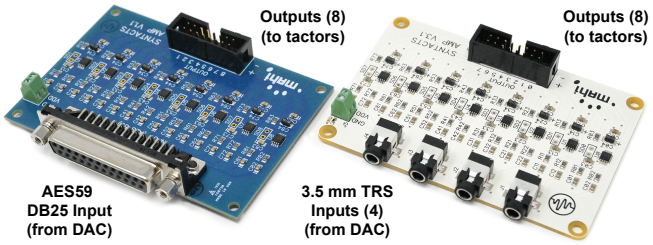

Figure 3: The Syntacts amplifier is an open-source fully differential, linear amplifier capable of driving eight vibrotactors with minimal noise. Two variants are available: one with a single AES-59 DB25 input for connecting to high-end audio devices such as the MOTU 24Ao, and one with four standard 3.5 mm TRS headphone inputs for connecting to general audio outputs or surround sound cards. Both require a 5V power supply, and output amplified signals through a universal 0.1" pitch header.

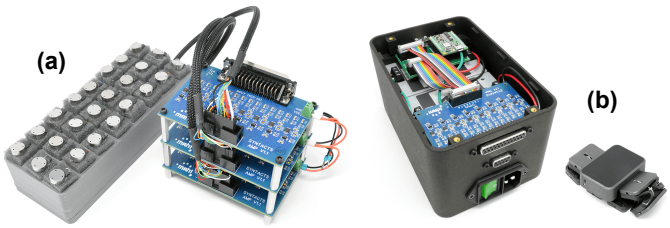

Figure 4: The Syntacts amplifier can be used in a variety of applications, ranging from dense tactile arrays (**a**) to wearable devices such as Tasbi (**b**). Designs for the tactile array are available online as a reference implementation.

Open-source designs for two variants of the amplifier, one with four 3.5 mm phone inputs and one with a standardized AES-59 DB25 connector, are available online along with manuals and data sheets. We provide packaged CAD files and BOMs for direct submission to turn-key PCB manufactures, where the boards can be built for roughly $50-100 USD depending on the quantity ordered and requested fabrication time. Alternatively, the PCB and components can be ordered separately and soldered by hand or in a reflow oven.

IV. Software for Audio-Based Control

Software is necessary both to interface audio devices and to synthesize and render waveforms. Many commercial GUI applications provide these features for the creation of music and sound effects. While some researchers have leveraged such software (particularly MAX MSP [15]) for vibrotactor control, they tend to be overly complex, lack features useful for haptic design, and are difficult to integrate with other applications programmatically. Though a number of haptic effect software GUIs and frameworks have been developed for commercial [19] or one-off, custom hardware [20], only a few examples of general purpose, audio-based vibrotactor software exist. One example is Macaron [11], a WebAudio-based online editor where users create haptic effects by manipulating amplitude and frequency curves. The software, however, is primarily focused on ease of design, and provides little in the way of device interfacing or integration with other code.

To this fill this void, we developed **Syntacts**, a complete software framework for audio-based haptics. Driven by the needs of both Tasbi [3] and MISSIVE [18], we have integrated a number of useful features, including:

- a user-friendly API that integrates with existing code
- direct access to external sound card devices and drivers
- flexible and extensive waveform synthesis mechanisms
- the ability to generate and modify cues in realtime
- spatialization of multi-channel tactor arrays
- saving and loading cues from a user library
- compatibility with existing file formats and synthesizers
- a responsive GUI for cue design and playback

Each point is further detailed in the following sections. Syntacts is completely open-source, with code and binaries for Windows and macOS freely available at: www.syntacts.org.

A. Syntacts API

Syntacts' primary goal is to provide a flexible, *code-oriented* interface that can be easily integrated with existing software and applications. The library is written in C and C++ to facilitate accessing low-level drivers and maximizing performance. Additionally, bindings are currently provided for C# and Python. The former is particularly useful for integrating Syntacts with Unity Engine for creating 3D virtual environments, while the latter allows for high-level scripting and interactivity (e.g., with Jupyter notebooks). Integration with other languages is possible via C shared library (i.e., DLL) loading, and additional languages may be officially supported in the future (e.g., a MATLAB interface would be useful to many academics). Code presented in this section is taken from the Python binding, but the native C++ API and C# binding are similar in their syntax and usage.

1) Interfacing Devices: Syntacts will interface with virtually any audio card on the commercial market. The API allows users to enumerate and select devices based on specific drivers, a feature typically reserved to professional commercial software. While Syntacts can open devices under any audio API, users should be mindful of the considerations discussed in Section 1, favoring low latency options such as ASIO. Library usage begins with creating an audio context, or Session. A Session opens communication with a requested audio device and starts an output stream to it in a separate processing thread.

```
# create an audio context
session = Session()

# enumerate connected hardware
for dev in session.available_devices:
    print(dev.index)         # e.g., 6
    print(dev.name)          # e.g., MOTU Pro Audio
    print(dev.max_channels)  # e.g., 24
    print(dev.api_name)      # e.g., ASIO
    ...                      # etc.

# open device 6 with 24 channels at 48 kHz
session.open(6,24,48000)
```

Listing 1: Querying hardware information and opening devices

2) Creating Effects with Signals: Vibration waveforms are represented abstractly by one or more Signals. Signal classes define a temporal sampling behavior and length, which may be finite or infinite. A variety of built-in Signals are available in Syntacts. For example, the classes Sine, Square, Saw, and Triangle implement typical oscillators with normalized amplitude and infinite duration, while Envelope and ASR (Attack, Sustain, Release) define amplitude modifiers with finite duration. Signals can be mixed using basic arithmetic. The act of multiplying and adding Signals can be thought of as an element-wise operation between two vectors. Multiplying two Signals yields a new Signal of duration equal to the shortest operand, while adding two Signals yields a new Signal of duration equal to the longest operand. Gain and bias can be applied to Signals with scalar operands as well.

In Listing 2 and Fig. 5, the Signals `sqr` and `sin` are implicitly of infinite length, while `asr` has a length of 0.3 s. Multiplying `sqr` by `sin` yields another infinite Signal with a 100 Hz square carrier wave, amplitude modulated with a 10 Hz sine wave (`sig1`). This Signal can further be given shape and duration by multiplication with `asr` to yield the finite Signal `sig2`. The Signal `sig3` represents another form of modulation through summation instead of multiplication. While the examples here only demonstrate passing scalar arguments to Signal constructors, some Signals can accept other Signals as their input arguments. For instance, it is possible to pass `sin` as the frequency argument to `sqr`'s constructor, yielding a form of frequency modulation. The modularity of the API allows users to create a wide variety of effects with minimal code. Syntacts can also be easily extended with custom user-defined Signals simply by creating classes which define the functions `sample` and `length`.

```
sqr = Square(100)          # 100 Hz square
sin = Sine(10)             # 10 Hz triangle
asr = ASR(0.1,0.1,0.1)     # attack, sustain, release

# basic examples mixing the Signals above
sig1 = sqr * sin
sig2 = sig1 * asr
sig3 = 0.5 * (sqr + sin) * asr

# play Signals on channel 0 and 1
session.play(0, sig1)      # plays until stopped
session.play(1, sig2)      # plays for 0.3 seconds
...
session.stop(0)            # stop sig1
```

Listing 2: Creating, mixing, and playing Signals

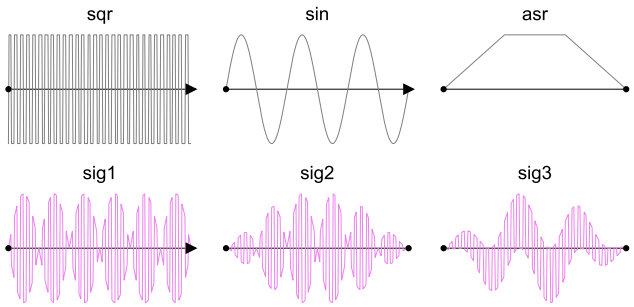

Figure 5: Signals created in Listing 2

3) Sequencing Signals: Multiple Signals can be concatenated or sequenced temporally to create patterns of effects using the insertion, or left-shift, operator. Consider the examples in Listing 3 and Fig. 6. First, two finite Signals `sigA` (0.3 s) and `sigB` (0.4 s) are created. Signal `sig4` demonstrates their direct concatenation, resulting in a 0.7 second long vibration where `sigB` is rendered immediately after `sigA`. Delay and pause can be achieved through the insertion of positive scalar operands, as shown in `sig5`. Inserting negative scalars moves the insertion point backward in time, allowing users to overlay or fade Signals into each other as in `sig6`. Sequences of Signals can also be sequenced as in `sig7`.

```
sigA = Sine(100) * ASR(0.1,0.1,0.1)      # 0.3 s
sigB = Sine(50) * ADSR(0.1,0.1,0.1,0.1)  # 0.4 s

sig4 = sigA << sigB                      # 0.7 s
sig5 = 0.1 << sigA << 0.2 << sigB        # 1.0 s
sig6 = sigA << -0.1 << sigB              # 0.6 s
sig7 = sig4 << sig5 << sig6              # 2.3 s

session.play(2,sig7)
```

Listing 3: Sequencing Signals in time

Figure 6: Sequenced Signals created in Listing 3

4) Spatialization and Realtime Modifications: In addition to playing Signals on discrete channels, multiple channels can be mapped to a normalized continuous 1D or 2D spatial representation with the Spatializer class. Similar to the Mango editor from Schneider et al. [7], users can configure a virtual grid to match the physical layout of a tactor array, and then set a virtual *target* coordinate and radius to seamlessly play and blend multiple tactors at once. Channel positions can be set individually or as uniformly spaced grids. Only channels within a *target* radius are played, and their volume is scaled according to a specified drop-off law (e.g., linear, logarithmic, etc.) based on their proximity to the target location. By moving the target location, for example, in a `while` loop or in response to changes in device orientation, developers can create sweeping motions and the illusion of continuous space with their tactile arrays (Listing 4, Fig. 7).

Other parameters, such as master volume and pitch, can be modified in realtime for Spatializers or individual channels. This offers developers the ability to move beyond playing discrete, pre-designed cues, to instead modifying continuous cues in response to conditions within the application. For example, consider the VR application in Fig. 10. In addition to pre-designed haptic effects that are triggered for specific events (such as button clicks), a continuous haptic effect is rendered when the player's hand is inside the fan air stream. Volume, the spatializer target, and pitch are changed based on hand proximity, wrist orientation, and the fan speed, respectively.

```
spatial = Spatializer(session)       # 2D Spatializer
spatial.create_grid(4,6)             # 4 rows X 6 cols
spatial.set_position(18,(0.1,0.8))   # move channel 18
spatial.radius    = 0.3              # effect radius
spatial.target    = (0.2, 0.1)       # target location
spatial.roll_off = 'linear'          # roll-off law
spatial.play(sig1)                   # play inf Signal

# modification in a loop
while condition:
    ...
    spatial.target = (x,y)
    spatial.volume = v
    spatial.pitch  = p

spatial.stop()
```

Listing 4: Spatializing tactor arrays and modifying parameters in realtime

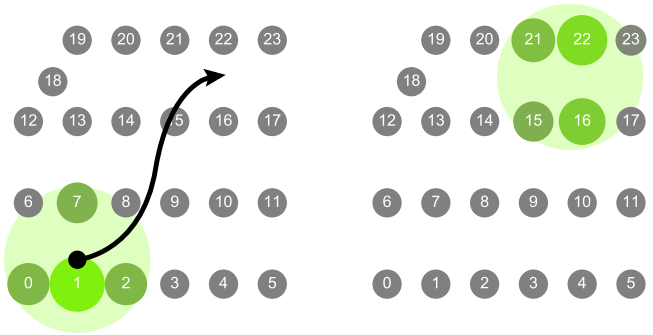

Figure 7: The Spatializer created in Listing 4

5) Saving and Loading Signals: User-created Signals can be saved to disk and reloaded at a later time using the functions `saveSignal` and `loadSignal`. The default file format is a binary representation of the serialized Signal. That is, instead of saving all individual audio samples, only the parameters needed to reconstruct the Signal at runtime are saved. This results in considerably smaller files which can be loaded more quickly on the fly than typical audio file formats. Nonetheless, Syntacts can still export and import WAV, AIFF, and CSV file formats for interoperability with existing haptic libraries.

B. Syntacts GUI

In addition to the raw APIs, Syntacts ships with a feature-rich GUI (Fig. 8). The GUI includes a drag-and-drop interface for designing Signals from built-in configurable primitives. The resulting Signal is immediately visualized to facilitate the design process. A track-based sequencer and spatialization editor are also included. Signals can be tested on a selected device's output channels, and then saved to the user's library for later use. Leveraging library features, users employ the GUI to rapidly tune haptic effects being loaded and played from Syntacts code in an separate application (e.g., iteratively tuning the effects for the buttons and knobs of the fan in Fig. 10). The GUI application is available as a precompiled executable or in source code format.

V. Comparison

In this Section, we evaluate Syntacts against two of the commercially available control options discussed in Section II: the EAI Universal Controller, and the TI DRV2605LEVM-MD evaluation board. Each controller was implemented with the manufacturer-recommended configuration so as to best compare them with the Syntacts framework.

A. Latency Benchmarking

Latency is a critical measure of a system's ability to render cues, especially for time sensitive applications like VR. For high-density tactile arrays, latency can increase with the number of channels simultaneously played since each subsequent channel adds more processing or transmission time. If multiple

Figure 9: Syntacts In Use - This figure demonstrates a real-world implementation of the Syntacts amplifier, where it has been used to drive two Tasbi haptic bracelets [3]. A professional grade audio device (MOTU 24Ao) is connected to two Syntacts amplifier boards that have been integrated into separate Tasbi control units. Amplifier output is transmitted to each Tasbi over a multi-conductor cable. Each Tasbi bracelet incorporates six Mplus 1040W LRA tactors radially spaced around the wrist, for a total of twelve utilized audio channels. The audio device interfaces with a host PC (not shown) through a USB connection.

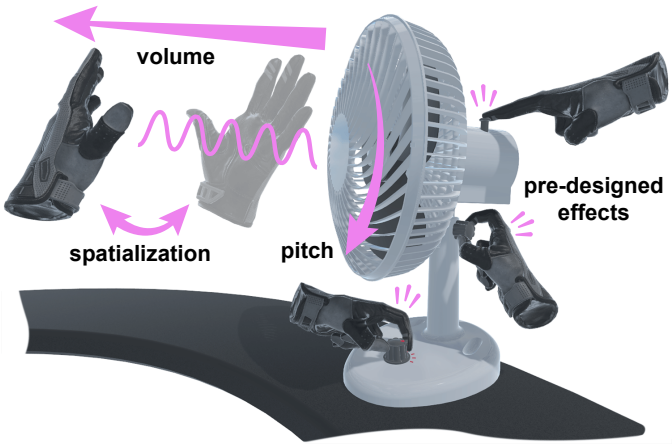

Figure 10: Syntacts In Use - Here, the C# binding of the Syntacts API is used in Unity Engine to provide haptic effects for a virtual fan interaction designed for the Tasbi setup shown in Fig. 9. Two usage paradigms are in effect. The first leverages pre-designed, finite Signals for knob detents (designed in the Syntacts GUI and loaded at runtime) and button contact events (created programmatically on-the-fly, parameterized by hand approach velocity). The second paradigm uses an infinitely long Signal for the fan air stream. The volume and pitch of this Signal are modified in realtime based on the user's hand location and the fan speed, respectively. One-dimensional spatialization is used to target only the tactors which are oriented toward the fan in a continuous fashion.

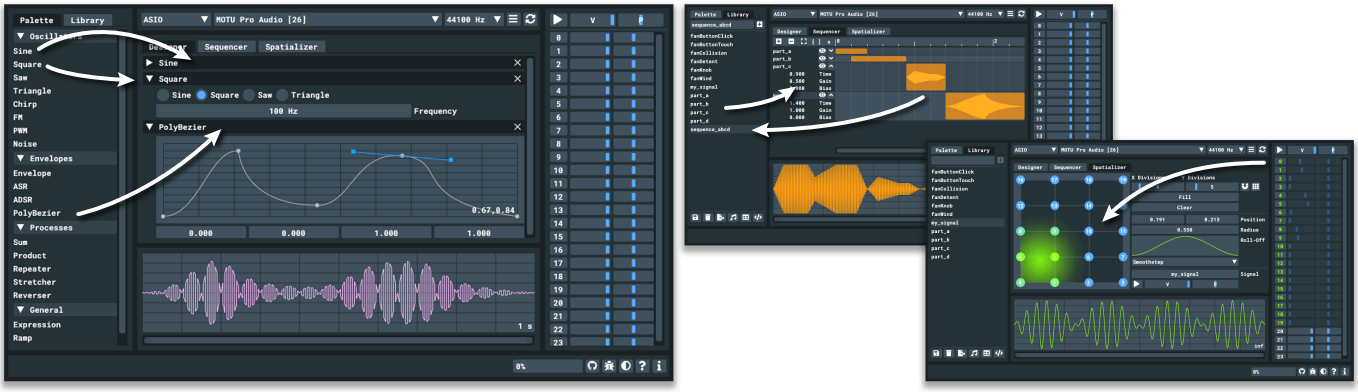

Figure 8: Syntacts GUI - The left-hand side demonstrates cue design. Users drag, drop, and configure Signals from the design Palette to the Designer workspace. The Signal is visualized and can be played on individual channels of the opened device. The right-hand side shows the GUI's track-based sequencer (background) and spatializer (foreground) interfaces. Once designs are complete, they can be saved and later loaded from the programming APIs.

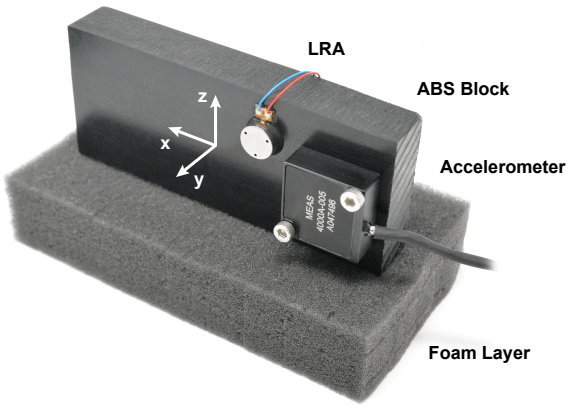

Figure 11: The testing apparatus used for all latency benchmarking. An Mplus ML1040W LRA was epoxied to a 100 g ABS block, and an accelerometer measured LRA induced vibrations along the y-axis. Latency was defined as the time from calling the software APIs to command vibration to the time at which 0.015 g of acceleration was measured.

channels are played at once, the last actuated channel may lag the first actuated channel by several milliseconds depending on the overall implementation. For this reason, we chose to benchmark latency as a function of the number of channels played at once.

We defined latency as the time from calling the functions to create and play a cue on $n = [1, 8]$ tactors until an appreciable acceleration (0.015 g) was measured on the last actuated tactor. To perform the test, we constructed an apparatus (Fig. 11) based on the factory testing rig for the Mplus ML1040W LRA vibrotactors that were used. An accelerometer (TE Connectivity 4000A-005) was attached perpendicular to gravity on a 100g block of acrylonitrile butadiene styrene (ABS). The block rested on a layer of polyurethane foam to mitigate external vibrations. A C++ testing application, also available online, controlled the experiments and ran 100 trials for each device. Data was collected with a Quanser QPID digital acquisition device polled at 50 kHz. All systems rendered a 178 Hz sine wave between $\pm$5V with a duration of 1,000 ms.

Syntacts software was configured to control a MOTU 24Ao under the ASIO driver API and a buffer size of 16, with power amplification being performed by the Syntacts amplifier board. Syntacts and EAI systems were controlled through their respective APIs, called directly from the testing application. As the datasheet for the EAI Universal Controller notes, it can only play four tactors at full amplitude simultaneously and its API imposes this limit, so its testing concluded there. Due to the nature of the TI chip, I^2C brokerage was required to interface with the testing application. We used an Arduino Uno for this purpose, under the assumption that it represented the most likely use case. The EAI and TI drivers were programmed to use manufacturer recommended methods to minimize cue latency.

Accelerometer data were reduced to find the mean and standard deviation of the latency for each system and number of channels played (Fig. 12). The Texas Instruments system has the highest latency for a single tactor, but does not increase latency through four tactors. After the fourth tactor, the average latency and standard deviation increase, possibly due to I^2C multiplexer components, but again stays constant after five tactors. The Arduino likely contributes most to this latency, but since it represents a very plausible implementation, we consider it a fair comparison. The EAI system has lower latency than the TI system for one and two tactors, but the latency linearly increases with number of channels played to greater than the TI system, and as noted cannot play more than four channels. The Syntacts system has significantly lower latency than either of the commercially available systems tested and does not seem to be a function of channels played, so the system could expand to larger tactor arrays without delays. Though not shown, we measured similar latency values for the MOTU 24Ao with 24 channels played simultaneously.

B. Overall Comparison

Whole-system comparisons of the vibrotactile control methods tested are summarized in Table I. The different programming APIs show the extent to which hardware can be integrated within software. The GUI column lists the different functionality of the included graphical user interfaces. Synthesizers are able to create cues, Sequencers have the ability to organize cues in time on one or more channels, and Spatializers allow users to specify the center of vibration for an array of tactors. The audio hardware listed only represents a small subset of the possible options, but as can been seen Syntacts allows users to select audio devices based on output needs and cost. For around $125 USD, researchers can interface a 7.1 surround sound card with Syntacts and the Syntacts amplifier to achieve a complete 8 channel setup comparable in performance to the $2,250 USD EAI Universal Controller. Though rendering more that 8 channels with audio comes at a cost, it can still be done for much less that the cost of multiple EAI controllers and is considerably more manageable than implementing an integrated circuit based design.

VI. Conclusions and Future Work

We have presented **Syntacts**, an open-source suite of vibrotactile software and hardware based on audio principles. The framework is purpose-built to control arrays of vibrotactors with extremely low latency. In addition, Syntacts includes a graphical user interface designed to synthesize and sequence cues, and spatialize them on tactile arrays. The **Syntacts**

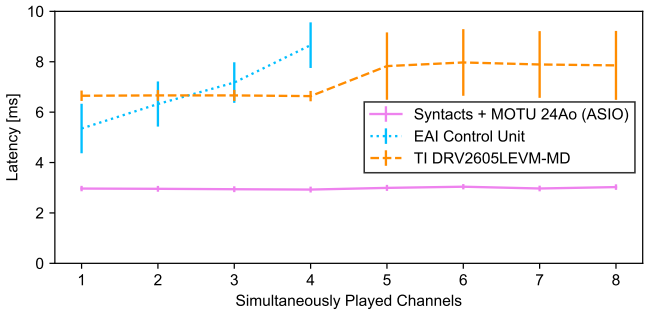

Figure 12: Latency as a function of channels rendered, measured as the time from software triggering to the detection of tactor acceleration. Only four channels are shown for the EAI control unit since this is its max.

Table I: Comparison of Tactor Control Methods Tested

Method	Interface	Open Source	API / Language	GUI	Hardware	Max. Channels	Avg. Latency (ms)	Approx. Cost (USD)
Audio	Syntacts	Yes	C, C++, C#, Python	Synthesizer Sequencer Spatialization	Headphone Jack	2	8.88	$75†
					SB Audigy RX7.1 (PCI-e)	8	4.22	$125†
					MOTU mk4 (USB)	12	5.20	$750†
					MOTU 24Ao (USB)	24	2.97	$1,225†
Controller	EAI	No	C	Synthesizer Sequencer	EAI Universal Controller	8‡	5.35	$2,250
IC	I^2C	Yes	N/A	Synthesizer	DRV2605LEVM-MD	8	6.65	$150

†Includes the cost of the number of Syntacts amplifiers (at $75 USD ea.) to accommodate the maximum available channels of the audio interface.
‡While the EAI Universal Controller supports eight channels, only four can be played simultaneously.

Amplifier easily integrates with the audio hardware to provide high-quality analog signals to the tactors without adding excess noise to the system. Importantly, neither Syntacts software nor the Syntacts amplifier are required by each other; users can choose to mix Syntacts software with their own amplifiers, or use the Syntacts amplifier with their own software. Finally, we benchmarked the Syntacts system against commercially available systems which showed that audio plus Syntacts is as, if not more, effective and flexible.

Syntacts is not without its limitations, and may not be the perfect tool for all researchers. For one, Syntacts requires a host PC and tethering hardware to audio interfaces. Therefore, Syntacts is not well suited for mobile or wireless haptic devices. Second, the Syntacts amplifier, while being compatible with large portion of commercial tactors, is not designed to power ERM or Piezo-actuators, and may have difficultly driving large and/or higher power actuators.

Future work for Syntacts involves both improvements on the usability of the software as well as understanding the use space more fully. In particular, immediate work will focus on extending realtime Signal modification features for VR applications. We aim to integrate more hapticly oriented tools as well, perhaps eventually favoring tactile perceptual models over audio centric concepts such as volume and pitch. Iteration on the API and GUI from user feedback would further increase the usability of the program. In closing, given the open-source nature of Syntacts, we welcome and hope that the haptics community will also contribute to its continued development.

REFERENCES

[1] S. Choi and K. J. Kuchenbecker, "Vibrotactile display: Perception, technology, and applications," *Proceedings of the IEEE*, vol. 101, no. 9, pp. 2093–2104, Sep. 2013.

[2] M. G. Carcedo *et al.*, "Hapticolor: Interpolating color information as haptic feedback to assist the colorblind," in *ACM Conf. on Human Factors in Computing Systems (CHI)*. ACM, 2016, pp. 3572–3583.

[3] E. Pezent *et al.*, "Tasbi: Multisensory squeeze and vibrotactile wrist haptics for augmented and virtual reality," in *IEEE World Haptics Conference (WHC)*. Tokyo, Japan: IEEE, July 2019.

[4] H. Culbertson *et al.*, "A social haptic device to create continuous lateral motion using sequential normal indentation," in *2018 IEEE Haptics Symposium (HAPTICS)*, 2018, pp. 32–39.

[5] Y. Konishi *et al.*, "Synesthesia suit: The full body immersive experience," in *ACM SIGGRAPH 2016 VR Village*, ser. SIGGRAPH '16. New York, NY, USA: Association for Computing Machinery, 2016. [Online]. Available: https://doi.org/10.1145/2929490.2932629

[6] A. Israr *et al.*, "Feel effects: Enriching storytelling with haptic feedback," *Transactions on Applied Perception (TAP)*, vol. 11, no. 3, pp. 11:1–11:17, Sep. 2014.

[7] O. S. Schneider *et al.*, "Tactile animation by direct manipulation of grid displays," in *Proceedings of the 28th Annual ACM Symposium on User Interface Software Technology*, ser. UIST '15. New York, NY, USA: Association for Computing Machinery, 2015, p. 21–30. [Online]. Available: https://doi.org/10.1145/2807442.2807470

[8] H. Seifi, *Crowdsourcing Haptic Data Collection*. Cham: Springer International Publishing, 2019, pp. 111–133.

[9] R. Rosenkranz and M. E. Altinsoy, "Tactile design: Translating user expectations into vibration for plausible virtual environments*," in *IEEE World Haptics Conference (WHC)*, July 2019, pp. 307–312.

[10] Y. Ochiai *et al.*, "Diminished haptics: Towards digital transformation of real world textures," in *Haptics: Neuroscience, Devices, Modeling, and Applications*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2014, pp. 409–417.

[11] O. S. Schneider and K. E. MacLean, "Studying design process and example use with macaron, a web-based vibrotactile effect editor," in *2016 IEEE Haptics Symposium*, April 2016, pp. 52–58.

[12] O. Schneider *et al.*, *FeelCraft: User-Crafted Tactile Content*. Tokyo: Springer Japan, 2015, pp. 253–259.

[13] M. Di Luca and A. Mahnan, "Perceptual limits of visual-haptic simultaneity in virtual reality interactions," in *2019 IEEE World Haptics Conference*, July 2019, pp. 67–72.

[14] C. Frisson *et al.*, "WebAudioHaptics: Tutorial on haptics with web audio," in *Web Audio Conference (WAC)*, 2016.

[15] A. Israr *et al.*, "Stereohaptics toolkit for dynamic tactile experiences," in *HCI International 2019 – Late Breaking Papers*, C. Stephanidis, Ed. Cham: Springer International Publishing, 2019, pp. 217–232.

[16] J. Chen *et al.*, "Feeling speech on the arm," in *Extended Abstracts of the 2018 CHI Conference on Human Factors in Computing Systems*, ser. CHI EA '18. New York, NY, USA: Association for Computing Machinery, 2018.

[17] C. M. Reed *et al.*, "A phonemic-based tactile display for speech communication," *IEEE Transactions on Haptics*, vol. 12, no. 1, pp. 2–17, Jan 2019.

[18] J. L. Sullivan *et al.*, "Multi-sensory stimuli improve distinguishability of cutaneous haptic cues," *IEEE Transactions on Haptics*, pp. 1–1, 2019.

[19] C. Swindells *et al.*, "Medium fidelity rapid prototyping of vibrotactile haptic, audio and video effects," in *IEEE Haptics Symposium*, Feb 2014, pp. 515–521.

[20] M. J. Enriquez and K. E. MacLean, "The hapticon editor: a tool in support of haptic communication research," in *Symposium on Haptic Interfaces for Virtual Environment and Teleoperator Systems*, March 2003, pp. 356–362.

Evan Pezent (S'16) received the B.S. degree from The University of Alabama, Tuscaloosa, AL, USA, in 2014 and the M.S. degree from Rice University, Houston, TX, USA, in 2017, both in mechanical engineering. He is currently a Ph.D. candidate in the Mechatronics and Haptic Interfaces Lab at Rice University. He received the NSF IGERT Fellowship in 2017. Since 2018, he has worked with Facebook Reality Labs developing wearable haptic devices and interactions for augmented and virtual reality.

Brandon Cambio (S'20) received the B.S. degree in Aeronautical Engineering from the United States Air Force Academy, Colorado Springs, CO, USA, in 2018, and the M.S. degree in Mechanical Engineering from Rice University, Houston, TX, USA, in 2020.

Marcia K. O'Malley (F'20) received the B.S. degree from Purdue University, West Lafayette, IN, USA, in 1996, and the M.S. and Ph.D. degrees from Vanderbilt University, Nashville, TN, USA, in 1999 and 2001, respectively, all in mechanical engineering. She is currently the Thomas Michael Panos Family Professor of mechanical engineering, of computer science, and of electrical and computer engineering with Rice University, Houston, TX, USA, and directs the Mechatronics and Haptic Interfaces Laboratory.