2021 43rd Annual International Conference of the
IEEE Engineering in Medicine & Biology Society (EMBC)
Oct 31 - Nov 4, 2021. Virtual Conference

A Platform for Virtual Reality Task Design with Intracranial Electrodes

Maurice Montag*, Courtnie Paschall*, Jeffrey Ojemann, Rajesh Rao, Jeffrey Herron, *Member, IEEE*

Abstract— **Research with human intracranial electrodes has traditionally been constrained by the limitations of the inpatient clinical setting. Immersive virtual reality (VR), however, can transcend setting and enable novel task design with precise control over visual and auditory stimuli. This control over visual and auditory feedback makes VR an exciting platform for new in-patient, human electrocorticography (ECOG) and stereo-electroencephalography (sEEG) research. The integration of intracranial electrode recording and stimulation with VR task dynamics required foundational systems engineering. In this work, we present a custom API that bridges Unity, the leading VR game development engine, and Synapse, the proprietary software of the Tucker Davis Technologies (TDT) neural recording and stimulation platform. To demonstrate the functionality and efficiency of our API, we developed a closed-loop brain-computer interface (BCI) task in which filtered neural signals controlled the movement of a virtual object and virtual object dynamics triggered neural stimulation. This closed-loop VR-BCI task confirmed the utility, safety, and efficacy of our API and its readiness for human task deployment.**

Clinical Relevance— **This work presents a novel Synapse-Unity API that enables new VR-supported clinical research in human subjects with ECOG and sEEG electrodes implanted for seizure localization.**

I. Introduction

Immersive virtual reality (VR) is an exciting new behavioral experimentation platform for human subjects research [1-5]. While VR has been key in many animal-model studies, the technology required to create world-scale, virtual reality experiments for humans has only recently become accessible. Using binocular image presentation, camera-and other sensor-based tracking, new VR head mounted displays (HMDs) can provide a complete three-dimensional reality with which humans can naturally interact and over which experimenters have near complete control. Already, portable and immersive VR has enabled new approaches to non-invasive electroencephalography (EEG) research in humans [1]-[5], and recent studies have noted the promise of VR-based experiments in outpatient subjects chronically implanted with the NeuroPace RNS device [6,7].

In contrast to scalp EEG, intracranial electrocorticography (ECOG) and stereo-electroencephalography (sEEG) electrodes offer a significant improvement in spatiotemporal signal resolution and enable direct electrical stimulation (DES) of the human brain. Intracranial electrodes are temporarily implanted in human patients to characterize seizures and identify foci of epileptogenic neural tissue that may be resected to treat seizure disorders. Such in-patient ECOG and sEEG therapeutic studies often incorporate over 200 electrodes from which simultaneous local field potentials (LFPs) can be recorded and precise electrical stimulation can be delivered.

In this paper, we present a systems engineering solution that bridges support for every VR HMD on the market and the Tucker-Davis Technologies neurophysiology suite, a premier neural recording and stimulation hardware platform. Our custom software enables easy-to-use plug-ins for VR-based experimental design, granting control of neural recording and neural stimulation parameters from within in an ongoing VR task. We demonstrate the functionality of our platform using an example closed-loop VR-BCI task. We also evaluate the efficiency and safety of our system during this task using benchtop hardware in a controlled laboratory setting.

II. Component Identification & System Integration

A critical first step towards an integrated VR and human neural recording and stimulation research platform was the identification of the VR and neural electrophysiology interface components. We selected Unity for its status as a mature VR development tool, ease of use, and plentiful tutorials that allow newcomers to create VR applications quickly using either a graphical user interface (GUI) or C# scripting [8]. Unity also provides plug-in support for a wide variety of virtual (VR), augmented (AR), and extended (XR) reality devices.

For our human electrophysiology platform, Tucker Davis Technologies (TDT) neural recording and neural stimulation devices were selected and approved for research use in humans by the Institutional Review Board (IRB). Synapse, the proprietary software and user-interface for TDT systems, enables real-time control of TDT hardware during an experiment, allowing query and update of signals processing, pulse generation, and stimulation parameters [9]. Synapse offers a RESTful API with bindings for Python, MATLAB, and C++, but does not support C# which is needed for Unity integration. Synapse also provides a UDP interface for minimal latency data transmission between an external

*Co-first authors who contributed equally to this work.

M. Montag is an undergraduate student in the University of Washington School of Computer Science and Engineering, Seattle, WA, USA (email: rmontag@uw.edu).

C. Paschall is a graduate student in the University of Washington Department of Bioengineering and a UW Reality Lab Fellow, Seattle, WA, USA (corresponding author, e-mail: copa2894@ uw.edu).

J. Ojemann is a Professor in the University of Washington Department of Neurological Surgery Seattle, WA, USA (email: jojemann@uw.edu).

R. Rao is a Professor in the University of Washington Department of Computer Science, Seattle, WA, USA (email: rao@cs.washington.edu)

J. Herron is an Assistant Professor in the University of Washington Department of Neurological Surgery, Seattle, WA, USA (email: jeffherr@uw.edu)

Research supported by the National Science Foundation (EEC-1028725) and the UW Reality Lab, Facebook, Google, Futurewei, and Amazon. The content is solely responsibility of the authors and does not necessarily represent the official views of any funding body.

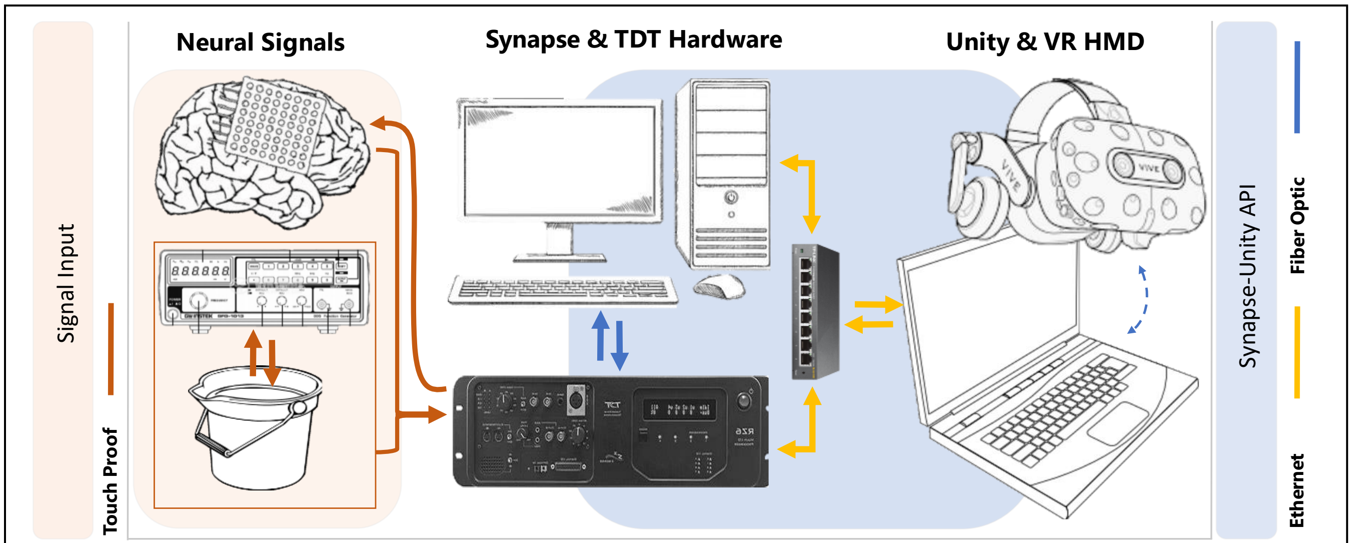

Figure 1. The hardware components of our testing and development setup. Signal input will come from the brain via implanted electrodes in human subjects, but this can be synthesized using electrodes, a function generator, and a saline tank in a laboratory space (depicted left and highlighted in red). The TDT recording hardware connects to the TDT computer hosting Synapse (middle column), while a separate task computer hosts Unity and supports the VR HMD hardware (right). Our developed API is deployed on the Unity-hosting computer and facilitates information transfer between the Unity environment and Synapse (blue). Various connections link these hardware components, including standard touch proof (red), fiber optic (blue), ethernet (yellow), and bluetooth (dashe blue). Wired and wireless HMD integration is supported within Unity.

computer and the neural signals processor, the RZ unit, for configurable use inside a Synapse-defined project. The bandwidth of the UDP interface is limited, however, so care must be taken when designing tasks to ensure that only necessary information, like synchronization and trigger cues, is passed over UDP. TDT documentation suggests that the maximum data throughput is 38.4 kBps, with 192 32-bit words being sent/received at 50 Hz.

For real-time Synapse and Unity interaction, we developed an API that manages the query and update of Synapse and Unity state variables during an ongoing experiment, while also governing data transfer between the TDT and VR task computers. Key design and evaluation metrics for our API included safety, within-experiment functionality, minimal lag, and ease of integration into standard Unity code.

A. The Synapse-C# interface

Integration of Synapse and Unity required development of a pure C# version of the provided Synapse API. Although we could have simply built a C# interface wrapping the existing API, we determined writing a new Synapse API in C# would be a more robust solution. This was due to Synapse's existing RESTful API being created with dynamically typed languages in mind, specifically Python and MATLAB. Although the existing C++ bindings of the RESTful API addressed the problem of dynamic typing by treating all data types as doubles, this solution did not allow for strict type safety. Human research demands fail-safe control of neural stimulation, so accurate data typing is requisite. Additionally, the existing bindings for Python, MATLAB, and C++ were distributed as a static Windows object library (.lib). We instead compiled our API in dynamic link library (.dll) format, for easier integration with Unity and other standard C# projects.

Finally, the existing RESTful API for Synapse was not object-oriented and mostly string-based, requiring unnecessarily clumsy syntax to work directly with Unity. In contrast, our C# Synapse API bindings facilitate the "getting" and "setting" of data inside Synapse in a simple, object-oriented fashion. This allows the code that interacts with Synapse to merge seamlessly with Unity code, eliminating awkward changes in syntax and allowing Unity developers to ignore the underlying complexity of the provided API, working instead with familiar object-oriented patterns. Our API also enables a more robust error handling system that eases the burden of interpreting error information returned from Synapse in Unity. Error handling in this context is extremely important. We use the C# concept of nullable value types to force the checking of potentially null data before use. For example, if a user tries to retrieve an API parameter that is not enabled in the API menu for the specific Synapse gizmo, the function will return a null value type. The user will have to check whether the value returned is not null before being able to use it unless they specifically use a cast to knowingly bypass the checking. Using this system instead of exceptions helps to eliminate the possibility of API generated crashes, leaving Synapse in an unwanted state during an experiment.

In Synapse, "Gizmos" are the functional units for recording and stimulation control. In our API, each Gizmo is given its own class, containing all API parameters and associated data. At startup, each Gizmo is instantiated with their name and type, and information is retrieved from the Synapse environment to determine the parameter boundaries of the configured Synapse experiment. These parameters and boundaries can then be called and modified from within the Unity environment. One of these gizmos is a generic purpose UDP interface which allow 32bit words transmitted to or from the RZ neural processor as an input or output to other gizmos. Our API also exposes the ability to read or write this RZ UDP port directly as a low latency input or output.

It is important to note that in this implementation data are being sent in the clear as JSON or UDP packets. Although TDT may address security concerns in the future, we linked

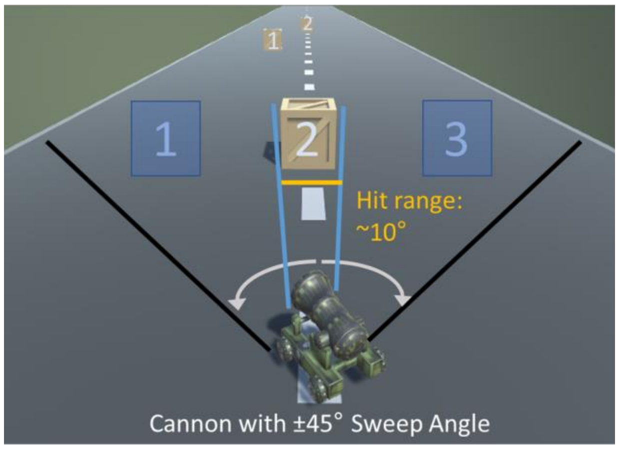

Figure 2. The 3D, in-game perspective of the cannon task. Band power in beta frequencies (10-30Hz) at one electrode is used to rotate the cannon in place with 90-degrees of swing. Two thresholds, low and high, define the beta power levels that control no rotation, counterclockwise rotation, or clockwise rotation of the cannon. Power estimates are completed by the TDT and imported each frame in the Unity task to update continuous cannon aiming. Projectiles are launched when a c ountdown timer reaches zero. If the projectile collides with the target box at one of three positions (for example, position 2, above), then the API triggers the TDT to deliver stimulation pulses to specified electrodes.

our Synapse and the Unity computers via a network (LAN) that was not connected to the external internet in order to keep patient behavioral and neural data secure (Fig 1).

B. *Validation using a benchtop Bi-directional VR-BCI Task*

To demonstrate the bidirectional capabilities of the Synapse-Unity API, we built a one-dimensional BCI control task in Unity (Fig 2). In this VR-BCI task, a cannon is rotated clockwise or counterclockwise according to the beta power (10-30Hz) recorded by a selected electrode. After between 10 and 15 seconds of beta-driven targeting, the cannon fires a cannonball. If this cannonball hits the depicted box target, the collision detected in Unity triggers a train of stimulation to be delivered by the TDT.

We synthesized real-time neural recordings in our laboratory environment in two ways. To test the coupling between beta-band power and cannon angle, we used a function generator and a bucket of saline solution set to mimic brain resistivity (Fig 1, left bottom). With the function generator, we introduced a complex voltage signal with a 2Hz sinusoidal power oscillation in the beta band. This signal was recorded by the TDT via a stereo-electroencephalography probe immersed in the saline solution (Fig 1). Using onboard digital signals processing hardware, the TDT band-passed the signal 10-30Hz and applied a Hilbert transform to extract the continuous power envelope, in real time. The cannon was seen to rotate as expected with this input (Fig 3, left). To evaluate our platform using realistic neural signals, we also utilized TDT's neural emulator software, Corpus, which simulates the spectral characteristics of human neural recordings. To clarify, Corpus is a neural emulator and does not replay actual neural recordings. Data were processed as before to extract beta-band power.

In both cases, two thresholds were selected based on standard deviations of beta power, creating low, medium, and high ordinal labels. For the Corpus emulator, these thresholds determined labels of $\beta_{LOW} \leq$ 2.2E-10 dBW, $\beta_{HIGH} \geq$ 3.2E-10 dBW, and 2.2E-10 < β_{MEDIUM} < 3.2E-10 dBW. Labels were used to update the rotation direction of the cannon every frame within Unity (60-90Hz): the ordinal label β_{LOW} triggered no cannon rotation, β_{MEDIUM} triggered counterclockwise rotation, and β_{HIGH} triggered clockwise rotation (Fig 3). If a cannonball collided with its appropriate target in Unity, stimulation was triggered. In our demonstration, stimulation was delivered as a train of constant current, biphasic pulses with pulse widths of 200us and a total train duration of one frame.

To assess the real-time performance of the developed software interface, we assessed system latency by measuring the delay from collision detection to stimulation onset. This was performed using a USB to serial port adapter to trigger TTL pulses at the start of each collision detected event. These TTL pulses were fed into an analog input channel and sampled at 24.4KHz by the TDT system such that the speed of the API in relaying collision, triggering stimulation, as well as any TDT hardware delays in executing the stimulation command could be measured.

III. Results

The tight coupling between the constructed signal's beta power in black and the change in the angle of the cannon in blue demonstrates the success of our task design and the rapid communication between the TDT recording hardware and the

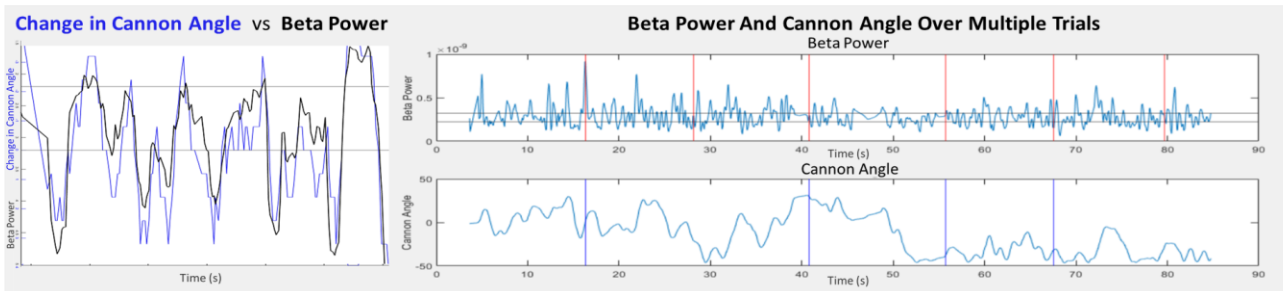

Figure 3. Three graphs depicting the relation between beta-band power and smoothed cannon angle. (left) The superimposed time series of the *change* in cannon angle (blue) and Beta power (black), showing clear and near-immediate correlation. (right top) A plot of continuous beta power over multiple aim-and-shoot trials, and (bottom right) a concurrent plot of the cannon angle over time. The horizontal black lines are the thresholds for counter- and clockwise cannon rotation. The vertical red lines indicate when cannon balls were fired at their targets, and vertical blue lines indicate when the target was hit and stimulation delivered.

Unity task environment (Fig 3, left). With the emulator, we collected over 550 trials of performance data to demonstrate the stability of the API. A trial is defined as one approximately ten second period aiming followed by one launch of a projectile. Trial number 552 is depicted below in Fig 3, right, as it had a high collision rate for an input that was effectively a random walk, in terms of task dynamics.

As seen in Fig 3, the cannon fires once every 10 to 15 seconds, marked by a vertical red line in the top graph showing beta power. The extended delay visible between the third and fourth cannon fire is the result of a delay between one batch of trials and another, a circumstance unique to the continuous stability test. If the cannon fires while it is at the appropriate angle to hit the target, the Unity experiment marks that trial as a successful collision, depicted in Fig 3 as a blue vertical line in the lower graph showing cannon angle.

The delay from collision detection to stimulation onset was shown to be small with little variation, on average 6.8 ms ± 1.7ms standard deviation across 217 collision-stimulation trials. Over 96.7% of delays were under 10ms, the maximum delay was 16.8ms, and the minimum delay 4.3ms.

IV. Discussion

By designing this integrated VR and human neurophysiology platform, we have provided experimentalists with a highly capable and flexible resource for developing virtual reality tasks, including bidirectional VR-BCI experiments. Our API permits rapid communication of task-relevant dynamics, such as ongoing band power measures or collisions within the virtual task, that are below visuohaptic and audiovisual perceptual delays [10,11]. While a researcher could instead use the newly released Python APIs from both Unity and TDT, the stability and temporal characteristics of this approach remain untested.

Our interface with the Unity game engine also enables support for augmented reality, mixed reality, mobile and PC-deployed applications. Additionally, Unity provides high-definition rendering, three-dimensional sound, and the integration of human physiologic data, like gaze-tracking, during task performance. Moreover, as an actively supported development platform, Unity is constantly integrating the latest hardware and improved software performance solutions. Our API links standard neural recording and neural stimulation hardware to this powerful engine for game-like behavioral task design.

It is worth emphasizing that support for Unity game development is immense, with easily accessible online documentation, free and paid tutorials, as well as many prebuilt assets. Basic Unity game design can be learned over the course of a few weeks, even for those entirely unfamiliar with game design or C# programming. Moreover, as Unity integrates new graphical scripting tools, the barrier to entry may be further reduced, allowing for those without any computer science background to create tasks quickly, creatively, and with minimal scripting. Linking human subject, invasive neural recording research to this design platform offers a new frontier of behavioral task design [7,12].

To that end, this API will soon be published on GitHub under a permissive license. Before public release, we intend to refine our documentation, add additional features for logging, and support new and parallel ways to ensure synchronized timing between Unity and Synapse. Our current API supports UDP-based synchronization as well as synchronization with analog signals such as USB bit output, audio cueing, and photodiode voltage recording. We also intend to release a general SynapseAPI experiment as a .synexp file that would demonstrate implementation of the API alongside a simple neural recording-based BCI task.

V. Conclusion

We designed and evaluated a VR development platform that integrates virtual reality devices with the hardware of in-patient, intracranial neural recording and neural stimulation research. We presented an example closed-loop VR-BCI task built with our system, as well as performance metrics relevant to closed-loop task design. We encourage communication from labs interested in implementing our Unity-Synapse platform and look forward to results of VR-based human intracranial research.

References

[1] S. Bouchard and A. Rizzo, “Applications of Virtual Reality in Clinical Psychology and Clinical Cognitive Neuroscience–An Introduction,” in *Virtual Reality for Psychological and Neurocognitive Interventions*, A. “skip” Rizzo and S. Bouchard, Eds. New York, NY: Springer New York, 2019, pp. 1–13.

[2] L. Bréchet *et al.* “First-person view of one’s body in immersive virtual reality: Influence on episodic memory,” *PLoS One*, vol. 14, no. 3, p. e0197763, Mar. 2019.

[3] K. Jahn *et al.* “Designing Self-presence in Immersive Virtual Reality to Improve Cognitive Performance—A Research Proposal,” in *Information Systems and Neuroscience*, 2020, pp. 83–91.

[4] A. Vourvopoulos *et al.* “Effects of a Brain-Computer Interface With Virtual Reality (VR) Neurofeedback: A Pilot Study in Chronic Stroke Patients,” *Front. Hum. Neurosci.*, vol. 13, p. 210, Jun. 2019.

[5] D. D. Georgiev *et al.* “Virtual Reality for Neurorehabilitation and Cognitive Enhancement,” *Brain Sci*, vol. 11, no. 2, Feb. 2021, doi: 10.3390/brainsci11020221.

[6] U. Topalovic *et al.* “Wireless Programmable Recording and Stimulation of Deep Brain Activity in Freely Moving Humans,” *Neuron*, vol. 108, no. 2, pp. 322–334.e9, Oct. 2020.

[7] A. Ramirez-Zamora *et al.* “Proceedings of the Seventh Annual Deep Brain Stimulation Think Tank: Advances in Neurophysiology, Adaptive DBS, Virtual Reality, Neuroethics and Technology *.” Front in Hum Neuro*, 2020.

[8] Tucker-Davis Technologies, “Synapse Manual.” https://www.tdt.com/docs/synapse/ (accessed May 2021).

[9] Unity Technologies, “User Manual 2020.3 (LTS)” https://docs.unity3d.com/Manual/UnityManual.html (accessed May 03, 2021).

[10] I. Vogels. “Detection of temporal delays in visual-haptic interfaces,” *Hum. Factors*, vol. 46, no. 1, pp. 118–134, Spring 2004. H.-Y.

[11] E. Zhou *et al.* “Audiovisual temporal integration: Cognitive processing, neural mechanisms, developmental trajectory and potential interventions,” *Neuropsychologia*, vol. 140, p. 107396, Mar. 2020.

[12] P. Kourtesis *et al.* “Guidelines for the Development of Immersive Virtual Reality Software for Cognitive Neuroscience and Neuropsychology: The Development of Virtual Reality Everyday Assessment Lab (VREAL), a Neuropsychological Test Battery in Immersive Virtual Reality,” Frontiers in Computer Science, vol. 1, p. 12, 2020.