A Flexible Multimodal Framework for Ecologically Valid Pain Research and Closed-Loop Clinical Translation

Shuangyi Tong, Pranav Mahajan, Danielle Hewitt, Sarah Schreiber, Yijia Yan, Rachel A. Crockett, Victoria S. Marks, Alexander L. Green, Timothy Denison, Ben Seymour

***Abstract*— Pain is a uniquely complex, subjective, and multidimensional human experience, continuously shaped by bodily state, environmental context, expectation, and action. Studying pain in a clinically meaningful way therefore demands experimental paradigms that are ecologically valid, multisensory, and capable of interactive, closed-loop control. Conventional neuroscience research setups, however, are typically tightly coupled to a fixed set of input/output signal modalities, which constrains the richness of behaviors that can be probed and the speed at which new paradigms can be assembled. To address this, we developed a flexible research and clinical platform built around a loosely coupled event-driven backbone. The framework integrates virtual reality (VR) to deliver immersive, context-rich environments, captures multimodal real-time data from wearable and implanted sensors, and delivers adaptive feedback such as cutaneous thermal or electrical stimulation and programmable neuromodulation. Drawing on several completed and ongoing pain studies, we demonstrate the platform's versatility and share the tools developed for its implementation. By lowering the engineering cost of assembling ecologically valid, multisensory pain paradigms, the platform provides a practical route toward personalized, closed-loop pain therapies in clinical settings.**

***Index Terms*— Pain, Ecological validity, Virtual reality, Closed-loop systems, Multimodal sensing, Neuromodulation**

The work was funded by Wellcome Trust (214251/Z/18/Z, 203139/Z/16/Z, 203139/A/16/Z, and WT223883/Z/21/Z), IITP (MSIT 2019-0-01371, RS-2023-00233251), JSPS (22H04998), and EPSRC (EP/W03509X/1). This work was also funded by the EPSRC and MRC Research Grant Scheme under the reference number UKRI1970. This work was also supported by the NIHR Oxford Health Biomedical Research Centre (NIHR203316) and the Royal Academy of Engineering. The views expressed are those of the author(s) and not necessarily those of the NIHR or the Department of Health and Social Care. For the purpose of open access, the author has applied a CC BY public copyright licence to any Author Accepted Manuscript version arising from this submission.

S. Tong (e-mail: shuangyi.tong@ndcn.ox.ac.uk), P. Mahajan, D. Hewitt, Y. Yan, and R. A. Crockett are with the Nuffield Department of Clinical Neurosciences, University of Oxford, Oxford, U.K. S. Schreiber and V. S. Marks are with the Institute of Biomedical Engineering, University of Oxford, Oxford, U.K. A. L. Green is with the Nuffield Department of Clinical Neurosciences and Nuffield Department of Surgical Sciences, University of Oxford, Oxford, U.K. T. Denison and B. Seymour are with the Nuffield Department of Clinical Neurosciences and also with the Institute of Biomedical Engineering, University of Oxford, Oxford, U.K.

I. Introduction

Pain is a uniquely complex human experience. It is not a direct readout of tissue damage, but a perceptual and behavioral state that the brain continuously infers from the state of the body, the surrounding environment, prior experience, expectation, and intended action [1], [2]. Because pain is shaped so strongly by context, it is notoriously difficult to capture in the reduced settings of conventional laboratory experiments, and even harder to modulate therapeutically in ways that generalize to a patient's daily life.

Evolutionary accounts of pain emphasize that the pain system is adapted to real-world behavioral contexts such as threat avoidance, body protection, recovery, and social signaling, rather than to the isolated stimulus-response tasks most often used in the laboratory [3]. This motivates a growing emphasis on *ecological validity*: the degree to which an experimental or therapeutic setting resembles the real environments in which behavior evolved and in which patients live. For pain research, ecological validity is not a cosmetic concern. Both the mechanisms of interest (e.g., threat appraisal, embodiment, avoidance learning, and effort-pain trade-offs) and the clinical outcomes of interest (e.g., chronic pain disability, rehabilitation engagement, and analgesic efficacy) are profoundly shaped by multisensory context and active behavior. Studying them with impoverished, isolated stimuli risks producing findings that do not translate to the clinic.

Achieving ecological validity in pain research requires three capabilities to operate in concert. First, the experimental setting must deliver rich, immersive, multisensory environments in which naturalistic behavior can unfold. Second, participant state must be measured continuously and across modalities, including movement, physiology, gaze, speech, and neural activity, so that context-dependent effects on pain can actually be detected. Third, the loop must be closed: stimulation, task parameters, or therapeutic feedback must be able to adapt in real time to what the participant is doing and feeling. Virtual reality (VR) has emerged as a particularly well-suited medium for the first of these requirements, because it allows precise experimental control over an environment that nonetheless feels embodied and behaviorally meaningful to the participant.

In practice, however, experimental infrastructure has been a major bottleneck to satisfying all three requirements at once.

Most neuroscience acquisition systems are tightly coupled to a fixed and limited set of input/output signal modalities. Adding a new sensor, a new form of stimulation, or a new model-based decision rule typically requires a bespoke engineering effort, and synchronizing them at low latency is harder still. As a result, the ecological paradigms one would ideally run for pain research, such as VR-based exploration with multisensory feedback, adaptive thermal or electrical stimulation contingent on behavior, and neurofeedback derived from wearable or implanted devices, remain the exception rather than the norm. The same bottleneck limits translation: a pipeline assembled for a single study rarely transfers to a clinical setting where personalized, closed-loop therapy is the goal.

Here we describe a flexible framework designed to address this bottleneck for pain research, and, by extension, for clinical translation. The framework is built around a loosely coupled backbone in which each device, sensor, or computational model connects as an independent client, publishes its data as time-stamped events to a central controller, and receives control signals back in the same standardized way. This decoupling is what provides the main practical benefit for experimenters and clinicians: new sensors and stimulators can be added with minimal engineering, and adaptive control logic written for one paradigm can be reused across many. Figure 1 summarizes this backbone. VR, motion capture, EEG, implanted neural devices, physiological sensors, speech input, peripheral and brain stimulation, and computational models of the participant's state all attach to a shared event loop.

Three features make this backbone particularly well matched to the needs of pain research. First, it makes ecologically valid task design substantially easier. Because modalities are not hard-wired to particular pipelines, researchers can assemble new multisensory VR paradigms, such as active exploration with thermal threat, biofeedback-driven relaxation, or effort-based exercise with adaptive nociceptive load, from a common toolkit rather than rebuilding from scratch for each study. Second, it supports multisensory integration at the level of real behavior: physiological, neural, behavioral, and contextual streams arrive at a single controller and can be combined to drive stimulation or to update models of the participant with low latency. Third, because the same framework that runs the experimental rig can also run an adaptive controller, it provides a direct translational path from laboratory paradigms to closed-loop clinical applications, such as VR-assisted rehabilitation, personalized stimulation therapies, and adaptive neuromodulation for chronic pain.

The remainder of this paper documents the framework and its use in pain research. We first give a high-level overview of the system architecture, followed by the specific implementations on the control and device sides. We then describe a series of pain studies in which the framework has been used, spanning behavioral, physiological, and model-based adaptive control [4]–[6]. The software is open-source and available in the Code Availability section. Throughout, our emphasis is on how the platform lowers the cost of building ecologically valid, multisensory pain paradigms and on its potential as a backbone for future personalized pain therapies.

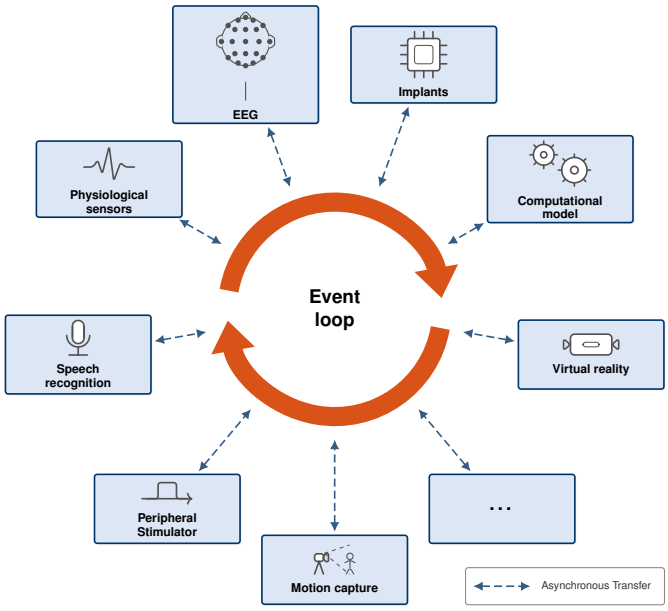

Fig. 1. Overview of the shared event-loop backbone. Heterogeneous sensing and stimulation modalities, including virtual reality, motion capture, electroencephalogram, implanted neural devices, physiological sensors, speech input, peripheral stimulation, and computational models of the participant, attach asynchronously to a single controller. This provides a common foundation for assembling ecologically valid, multisensory pain paradigms. EEG: Electroencephalogram

II. System Implementation

A. Implementation Overview

Our system is designed for research and clinical applications, which are typically directed by a researcher or clinician. Accordingly, we adopted a client-server model in which each device acts as a client communicating exclusively with a central controller (server). The controller monitors all network traffic and manages every device registered in the system.

To establish a connection, a device first sends a device descriptor to the server. This descriptor includes a unique device ID and specifies the formats for the data it sends and receives. Data sent from a device to the server is termed *report data*, while data sent from the server to the device is termed *control data*. Upon receiving data, each side is required to send a status code of up to 4 bytes in response, confirming the transmission. Communication is bidirectional and concurrent. Figure 2 depicts the transmission sequence for both report and control data.

In a typical neuroscience experiment, a wide range of devices transmit data across very different modalities. To balance development flexibility with performance, we designed a communication protocol that accommodates both high-resource and low-resource devices. For devices with ample computing resources, JavaScript Object Notation (JSON) is used as the data interchange format, because it is lightweight, versatile, and well-supported across programming languages. For devices with limited computing resources, such as microcontrollers, data can instead be transmitted and received in a compact binary format to avoid the overhead of serialization. In that case, the device must declare the binary field layout

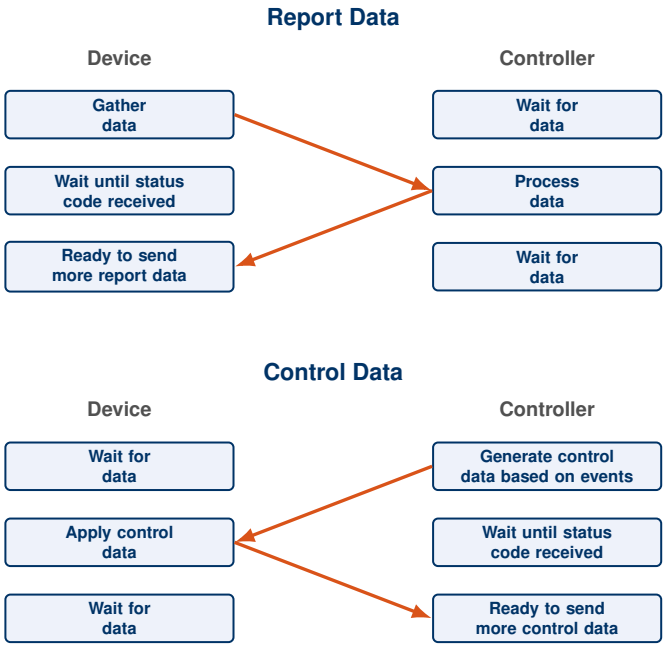

Fig. 2. Transmission sequence for the two event directions. Report-data and control-data transmission can proceed concurrently, but within each direction the sender must receive the status-code acknowledgment from the previous transmission before sending the next one.

TABLE I
COMPARISON OF DATA TRANSMISSION TYPES SUPPORTED BY THE PROTOCOL

Types of transmission	Descriptor specifies binary field layout	Supported data types	Requires additional library
High resources	No	JSON supported data types: object, array, number, string, boolean, null	JSON serialization/ deserialization
Low resources	A JSON array containing data field name and types	Unsigned 16- and 32-bit integers, encoded little-endian	No

in its descriptor so that the controller can decode the payload correctly. Table I summarizes the two transmission types.

B. Control-side Implementation

In this section, we illustrate three key features of our control-side (server-side) program implementation.

1) Event-driven Task-specific Control: We implemented the control-side program in the Node.js runtime environment, which is built on a single-threaded cooperative event loop with asynchronous I/O and provides comprehensive event-driven style APIs [7]. Data transport and addressing are handled using the TCP/IP protocol. Upon receipt, each incoming packet is decoded into a source identifier and a structured dataframe, which are then passed as two arguments to a designated function named `actionFunction`. New task developers are expected to override this function with custom logic for specific tasks. This design provides a uniform control entry point, encapsulated within a single function call. The `actionFunction` uses the source identifier to dispatch the dataframe to the appropriate processing branch. To generate feedback, control commands are written asynchronously by pushing them into a dedicated data queue.

Algorithm 1 Task-Specific Control Logic

Definitions:
ID_CONST_1, ID_CONST_2, ID_CONST_3 : Constant unique identifiers for source/target devices
$\mathcal{S}_{init}$: Initial state configuration
$\mathcal{D}$: Incoming dataframe

$state \leftarrow \mathcal{S}_{init}$
$d_{id1} \leftarrow$ ID_CONST_1
$d_{id2} \leftarrow$ ID_CONST_2
$d_{id3} \leftarrow$ ID_CONST_3

function ACTIONFUNCTION($d_{in}, \mathcal{D}$)
  if $d_{in} = d_{id1}$ **then**
    $cmd \leftarrow$ CALCCONTROL($state, \mathcal{D}$) `// Process data & generate feedback`
    ASYNCSEND(d_{id2}, cmd) `// Dispatch to target device`
  else if $d_{in} = d_{id3}$ **then**
    ...
  end if
end function

The `actionFunction` is overridden by providing a task-specific JavaScript file in which stateful variables are defined in the enclosing scope and captured in the function's closure. Algorithm 1 illustrates the typical structure of such a task-specific control script.

2) Real-time Data Visualization: Within its descriptor file, a device can specify the visualization format for its reported data. When this visualization format is properly specified, the control-side program can render the data in real-time (Figure 3A). Task-specific scripts may also contain important state variables that researchers need to monitor in real-time. To facilitate this, we created a lightweight JSON-based configuration schema that allows researchers to specify which variables should be displayed in real-time. The schema mainly specifies how data can be converted between displaying string variables and internal data types. The schema is expressed as a subset of JSON and must be placed at the beginning of a script to ensure it is processed correctly by the program.

3) Ad Hoc Control By The Researcher: In addition to the algorithmic commands generated by a script, it is often necessary in practice for researchers to perform ad hoc control of the task directly, for example, to change the current trial number. This capability was also implemented using the same JSON-based configuration schema, which preserves flexibil-

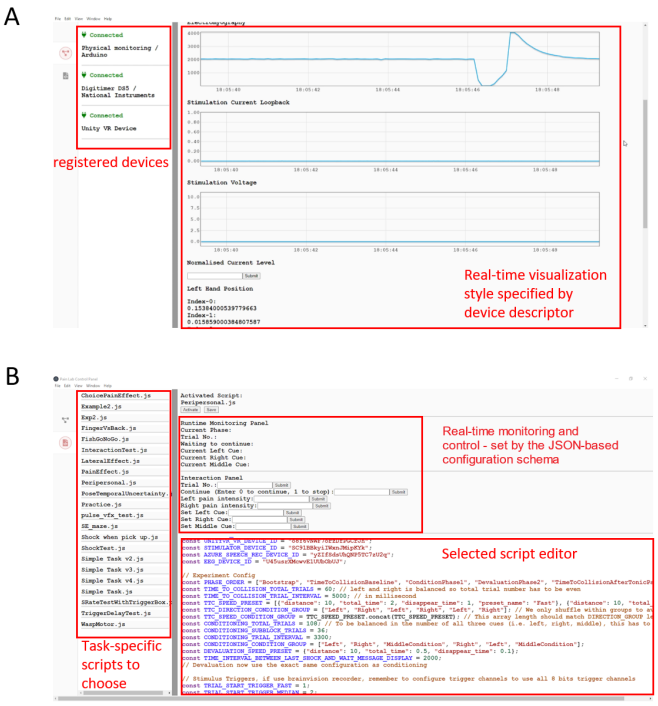

Fig. 3. Main views of the control-side program developed with Electron. **A**, Left panel: list of registered devices. Right panel: real-time visualization of report data. **B**, Left panel: Task-specific control scripts available to choose. Right panel: top, script management (activate and save), middle, script state variable visualization and control, bottom, script editor.

Fig. 4. A zoom-in view of the JSON-based configuration schema and the runtime monitoring and control panel.

ity while enabling researchers to maintain sufficient manual control during the script's execution (Figure 3B). Using this schema, researchers specify the target variable to modify and a corresponding function name that converts an input string into the appropriate data type (Figure 4). This ad hoc control mechanism modifies the script's runtime state directly, bypassing the main event system to prioritize changes applied by researchers.

C. Device-side Implementation

In practice, the framework integrates a diverse array of hardware and software components, varying widely in event transmission rates and computational capacity. For example, a VR environment demands continuous, high-frequency bidirectional state synchronization, whereas a microcontroller-based thermal stimulator may only require a single control command followed by a brief sequence of temperature reports. To standardize the integration of such heterogeneous systems, we categorize device-side implementations based on the temporal dependence between data channels. Temporal dependence describes the degree of causal coupling between these channels, specifically, whether a transmission is blocked by or directly triggered by a reception. Analogous to telecommunications, dependent implementations function similarly to half-duplex links, while independent implementations require the decoupled concurrency of a full-duplex setup. The following sections present specific algorithmic patterns for these categories complying with our system protocol.

1) Unidirectional Devices: The first class of devices is characterized by a single direction of event flow and is commonly encountered in neuroscience research. Physiological sensors are typical examples. A practical instance is a physiological sensor implemented on an Arduino Nano 33 IoT to collect skin conductance response (SCR) and photoplethysmography (PPG) data. Because event traffic flows in only one direction, the notion of temporal dependence does not apply here; the device behaves as a logically simplex reporter at the event level. To support implementation on resource-constrained hardware, our data protocol permits transmitting high-sample-rate physiological data directly to the control-side program by declaring the binary field layout in the descriptor. High-rate data can therefore be collected and wirelessly transmitted using common, low-cost IoT microcontrollers with a built-in network stack, without any additional library. The corresponding control flow is given in Algorithm 2.

2) Temporally Dependent Devices: Parallel execution is often beneficial when report data and control commands need to be transmitted simultaneously, in order to avoid data loss. When the two are causally coupled, however, they can be transmitted sequentially without conflicts or noticeable delay. An example is our thermal stimulator controller, which runs on a single-core embedded computer. The device waits for a stimulation command from the control-side program and then reports a short sequence of temperature samples following that stimulation, making the exchange effectively half-duplex at the event level. Algorithm 3 summarizes this control flow.

3) Temporally Independent Devices: When one channel requires continuous transmission while the other has a low event rate but demands low latency, an asymmetric, temporally independent pattern arises. Multi-threading with standard syn-

Algorithm 2 Unidirectional Device Control Logic

Definitions:
$flag_{send}$: Boolean permission state (initially **true**)
$\mathcal{D}_{out}$: Collected dataframe to be reported

while true do
  $\mathcal{D}_{out} \leftarrow$ TRYCOLLECTDATA
  if not $flag_{send}$ **then**
    $flag_{send} \leftarrow$ CHECKSERVERRESPONSE // Poll server for ack
  else
    if $\mathcal{D}_{out} \neq$ **null then**
      SENDREPORTDATA($\mathcal{D}_{out}$) // Transmit dataframe
      $flag_{send} \leftarrow$ **false** // Wait for ack before next send
    end if
  end if
  $flag_{send} \leftarrow$ CHECKSERVERRESPONSE // Poll server for ack
end while

Algorithm 3 Temporally Dependent Device Control Logic

Definitions:
R_{srv} : Received data from control server
C_{stat} : Status code derived from control instructions
$\mathcal{D}_{out}$: Local sensor dataframe

while true do
  $R_{srv} \leftarrow$ READSERVERDATA // Blocking wait for incoming packet
  if ISSTATUSCODE(R_{srv}) **then**
    CHECKSTATUSCODE(R_{srv})
  else // Packet contains control data
    $C_{stat} \leftarrow$ HANDLECONTROLDATA(R_{srv})
    SENDSTATUSCODE(C_{stat})
    $\mathcal{D}_{out} \leftarrow$ COLLECTDATA
    SENDREPORTDATA($\mathcal{D}_{out}$)
  end if
end while

chronization primitives provides an efficient implementation. One example is the control of an electrical stimulator: the device continuously reports its DC voltage and current while concurrently waiting for control commands to initiate stimulation. An EEG device fits the same pattern when conventional trigger signals are required, since it continuously streams data to the controller while occasionally accepting triggers that must be aligned with the data stream. Algorithm 4 illustrates this control flow, using a semaphore and a mutex to obtain a race-condition-free implementation.

D. Technical Implementation Choice

A number of open-source and closed-source data acquisition platforms are available for neuroscience research, but the primary focus of most of them is data acquisition itself. For studies that require only acquisition, our framework may offer little advantage over established software, particularly

Algorithm 4 Temporally Independent Device Control Logic (with Mutex)

Definitions:
S_{sem} : Counting semaphore instance
M_{mtx} : Mutex instance
B_{ctrl} : Shared memory buffer for control data
R_{srv} : Received data from control server
C_{stat} : Status code derived from control instructions
$\mathcal{D}_{out}$: Local sensor dataframe

$S_{sem} \leftarrow$ SEMAPHORE
$M_{mtx} \leftarrow$ MUTEX
$B_{ctrl} \leftarrow$ INITDATABUFFER
$\mathcal{D}_{out} \leftarrow$ COLLECTDATA // Bootstrap: collect and send
SENDREPORTDATA($\mathcal{D}_{out}$) // the first report to start the ack chain

— REPORT DATA THREAD —
while true do
  $R_{srv} \leftarrow$ READSERVERDATA
  if ISSTATUSCODE(R_{srv}) **then**
    $is\_ok \leftarrow$ CHECKSTATUSCODE(R_{srv})
    if is_ok **then**
      $\mathcal{D}_{out} \leftarrow$ COLLECTDATA
      SENDREPORTDATA($\mathcal{D}_{out}$)
    end if
  else // Must be control data
    LOCK(M_{mtx})
    COPYMEMORY(B_{ctrl}, R_{srv})
    UNLOCK(M_{mtx})
    SEMAPHOREINCREMENT(S_{sem})
  end if
end while

— CONTROL DATA THREAD —
while true do
  SEMAPHOREDECREMENT(S_{sem})
  LOCK(M_{mtx})
  $C_{stat} \leftarrow$ HANDLECONTROLDATA(B_{ctrl})
  UNLOCK(M_{mtx})
  SENDSTATUSCODE(C_{stat})
end while

given their broader hardware compatibility. Lab Streaming Layer (LSL), for example, is a powerful open-source tool that uses a custom TCP-based wire protocol with XML stream metadata and timestamp-based synchronization for robust data collection [8]. While an event-driven system could be layered on top of existing application-layer streaming protocols such as LSL, or on top of general-purpose messaging protocols like MQTT that are widely used in industrial IoT deployments, our implementation instead targets the specific requirements of closed-loop adaptive control research while remaining flexible in its hardware requirements. For instance, our lightweight protocol can be readily implemented on embedded devices with limited memory and storage. The key distinction is

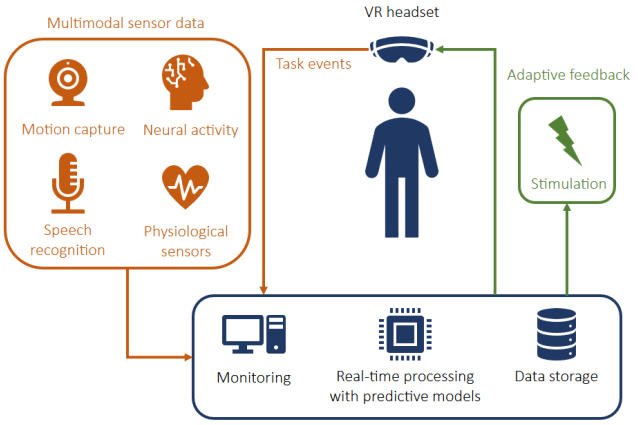

Fig. 5. Schematic of the ecologically valid virtual reality (VR) system for pain research. Components are grouped into three categories: sensing devices (orange) that send report data from the participant to the controller, feedback devices (green) that carry control data from the controller back to the participant, and control-side infrastructure (blue) that runs the adaptive control logic.

one of design orientation: LSL is optimized for data acquisition with timestamp-based synchronization, whereas our framework centers on event-triggered computation within a cooperative single-threaded control flow, lowering the engineering cost of implementing closed-loop adaptive paradigms across heterogeneous devices. This architecture simplifies the integration of additional sensor modalities, feedback channels, and control algorithms across diverse hardware, making it well suited for developing novel ecologically valid tasks and adaptive control paradigms.

III. Adaptive Control with the Event-driven Architecture

Over the past several years, we have used this event-driven system to run a range of ecologically valid pain research studies. In this section, we describe the experimental setup common to these studies and present three representative use cases, organized by the class of event flow that drives the adaptive control: behavioral events in the virtual environment, real-time neural activity from an implanted device, and outputs of a predictive model fit to multimodal data. A schematic of the overall setup, covering sensing, stimulation, and control-side infrastructure, is shown in Figure 5.

A. Adaptive control by behavioral events

In an ecologically valid pain experiment, a natural control flow is to deliver stimuli that are contingent on what the participant encounters and does within a virtual environment. VR is well suited to this pattern because it exposes behavioral data directly through the game engine and generates high-frequency, bidirectional traffic on both the report and control channels. A simple and effective strategy is to report a full behavioral snapshot on every render frame. Such a snapshot typically contains the spatial location of each tracked body part, every interaction initiated by the participant, relevant environmental changes, and ancillary signals collected by the game engine, such as eye tracking [9]. A practical sufficiency check is whether the virtual world and all interactions can be exactly replayed from the logged stream; if they can, the controller has enough information to make adaptive decisions in real time and offline.

Given such a stream, the control-side program has the context it needs to apply adaptive stimulation. For phasic pain, a brief peripheral stimulus can be triggered conditionally, for example when the participant makes an incorrect choice, picks up a designated dangerous object, or executes a penalized movement. For tonic pain, sustained control follows the same pattern: a continuously rising temperature can be delivered via cutaneous heat stimulation as the participant reaches further into a pool of virtual hot water. Using the same event-driven substrate, we have been able to explore a wide range of pain stimulation paradigms under different behavioral conditions within an ecologically valid context [4]–[6].

B. Adaptive control by neural response

A more demanding case of adaptive control arises when the input driving the VR experience is recorded directly from the brain. We used the framework to implement a real-time neurofeedback task for a patient implanted with a Picostim DyNeuMo-1 deep brain stimulation (DBS) device for chronic central post-stroke pain [10]. Local field potential (LFP) signals from the patient's periaqueductal grey (PAG) lead were streamed from the DBS platform to the central controller as a standard report-data client. On the control side, a short task-specific script estimated the instantaneous spectral power within a patient-specific frequency band from the most recent 512 samples with a Hann window, and dispatched the resulting value asynchronously to the VR client. The VR client, built in Unity and rendered on a Meta Quest Pro head-mounted display, mapped this value to the radius of a single 3D sphere shown to the patient (Figure 6).

Before each session, the mapping from band power to sphere size was calibrated per patient. A short passive-viewing epoch defined the lower anchor, and a brief attend-to-pain epoch defined the upper anchor, with clipping applied so that the visual feedback remained within the calibrated range. The patient was then guided through blocks in which they were asked either to attend to their pain, to apply their own imagery-based distraction strategy while watching the sphere, or to apply the same strategy while viewing only a fixation cross. Throughout each session, LFP samples, stimulation state, and VR task events were all routed through the shared event loop, so that the patient's neural activity drove the visual feedback with low latency and every block of data remained synchronized with the corresponding task events for offline analysis.

From a systems standpoint, this case illustrates two properties of the framework that are particularly relevant to clinical translation. First, the device integration layer is agnostic to whether the input is a low-cost peripheral sensor or an implanted research-grade neurostimulator: the DBS device attaches as a report-data client in exactly the same way as any other sensor, and the same control-side infrastructure

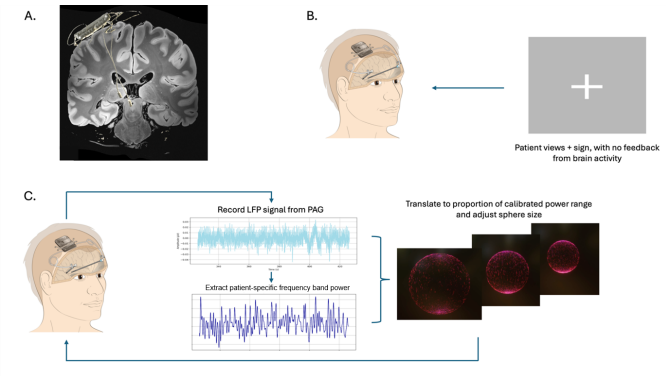

Fig. 6. Real-time neurofeedback driven by local field potentials (LFPs) implemented on the framework. LFPs recorded from the periaqueductal grey via a Picostim DyNeuMo-1 deep brain stimulation device are streamed to the central controller as standard report data, band-power within a patient-specific frequency range is estimated on the most recent samples, and the resulting value drives the radius of a 3D sphere rendered in a virtual reality scene on a head-mounted display. The same shared event loop carries task events and stimulation state, so that neural activity, visual feedback, and behavioral context remain aligned during a session with a real patient. PAG: Periaqueductal gray

handles feature extraction and feedback. Second, the entire patient-facing pipeline, spanning signal streaming, band-power estimation, visual mapping, and event logging, is expressed as a short task-specific script on top of the shared controller. This makes it practical to iterate on a neurofeedback paradigm at the bedside with a real patient, rather than rebuilding bespoke middleware for every new study.

C. Adaptive control by predictive models

Because the event-driven design naturally integrates data across modalities, a particularly powerful use case is model-based control driven by the combined stream. One scenario is a task in which experimental contingencies are manipulated dynamically. By continuously fitting an interpretable model of the participant's brain or behavior to real-time data, the control algorithm can target specific model parameters for promotion or suppression [11]; concurrently, the model refines its predictions as new physiological and behavioral data arrive.

The same predictive-modeling approach also extends naturally to outcome-oriented control. Because exercise plays an important role in the management of chronic pain [12], VR experiences are a promising adjunct for encouraging physical activity in patients living with pain [13]. To this end, we have proposed a model that provides a combined valuation over phasic pain, tonic pain, and physical state [5]. Coupled with an online estimate of physical effort and pain severity, such a model can in principle be used to adaptively adjust exercise intensity in pursuit of an optimal therapeutic balance.

The control algorithm itself can range from a simple linear PID controller to a non-linear neural-network approximator [14]. In many practical cases, the model's update rate is lower than the rate of incoming events, but the event-driven design handles this mismatch naturally: the model runs in a separate process or on a dedicated device at its own cadence, and when a new prediction is ready it is published as an event that the control algorithm consumes like any other input to update the VR experience or the stimulation accordingly.

IV. Conclusion

In this work, we have presented a flexible, loosely coupled framework that lowers the engineering cost of building ecologically valid, multisensory paradigms for pain research. By attaching diverse sensors, stimulators, and computational models to a shared event-driven backbone, the framework unifies multimodal data streams with very different sampling rates into a common representation, and supports low-latency adaptive control without bespoke integration work for each new study. We have demonstrated this across multiple pain studies spanning behavioral, neural, and model-based adaptive control, and we have shared a user interface developed iteratively alongside researcher and clinician feedback to keep the system usable in practice.

Pain is a uniquely context-dependent experience, uniquely bound up with movement and topographic embodiment. We believe the clearest path to clinically useful findings runs through paradigms that embrace this context. Looking forward, a key objective is to use this framework to deliver personalized, closed-loop pain therapies, for example, VR-based rehabilitation adapted in real time to physiological state, or adaptive neuromodulation driven by biomarkers extracted from wearable and implanted devices. By providing a common backbone that connects the laboratory and the clinic, the framework is intended to support not only more ecologically valid pain research, but also its translation into personalized treatments for patients living with pain.

References

[1] B. Seymour, "Pain: A precision signal for reinforcement learning and control," *Neuron*, vol. 101, no. 6, pp. 1029–1041, 2019. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0896627319300820

[2] P. Mahajan, P. Dayan, and B. Seymour, "Homeostasis after injury: How intertwined inference and control underpin post-injury pain and behaviour," *PLOS Computational Biology*, vol. 22, no. 1, pp. 1–17, 01 2026. [Online]. Available: https://doi.org/10.1371/journal.pcbi.1013538

[3] B. Seymour, R. J. Crook, and Z. S. Chen, "Post-injury pain and behaviour: a control theory perspective," *Nature Reviews Neuroscience*, vol. 24, no. 6, pp. 378–392, Jun 2023. [Online]. Available: https://doi.org/10.1038/s41583-023-00699-5

[4] P. Mahajan, S. Tong, S. W. Lee, and B. Seymour, "Balancing safety and efficiency in human decision making," Jun. 2025. [Online]. Available: http://dx.doi.org/10.7554/eLife.101371.2

[5] S. Tong, T. Denison, D. Hewitt, S. W. Lee, and B. Seymour, "Phasic and tonic pain serve distinct functions during adaptive behaviour," Aug. 2025. [Online]. Available: http://dx.doi.org/10.7554/eLife.107911.1

[6] D. Hewitt, S. Tong, S. Schreiber, and B. Seymour, "Tonic pain modulates neural correlates of associative phasic pain memories," *PAIN*, 2026. [Online]. Available: https://journals.lww.com/pain/fulltext/9900/tonic_pain_modulates_neural_correlates_of.1111.aspx

[7] S. Tilkov and S. Vinoski, "Node.js: Using javascript to build high-performance network programs," *IEEE Internet Computing*, vol. 14, no. 6, pp. 80–83, 2010.

[8] C. Kothe, S. Y. Shirazi, T. Stenner, D. Medine, C. Boulay, M. I. Grivich, F. Artoni, T. Mullen, A. Delorme, and S. Makeig, "The lab streaming layer for synchronized multimodal recording," *bioRxiv*, 2025. [Online]. Available: https://www.biorxiv.org/content/early/2025/07/14/2024.02.13.580071

[9] I. B. Adhanom, P. MacNeilage, and E. Folmer, "Eye tracking in virtual reality: a broad review of applications and challenges," *Virtual Real*, vol. 27, no. 2, pp. 1481–1505, Jan. 2023.

[10] M. Zamora, R. Toth, F. Morgante, J. Ottaway, T. Gillbe, S. Martin, G. Lamb, T. Noone, M. Benjaber, Z. Nairac, D. Sehgal, T. G. Constandinou, J. Herron, T. Z. Aziz, I. Gillbe, A. L. Green, E. A. Pereira, and T. Denison, "DyNeuMo Mk-1: Design and pilot validation of an investigational motion-adaptive neurostimulator with integrated chronotherapy," *Experimental Neurology*, vol. 351, p. 113977, 2022. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0014488622000024

[11] J. H. Lee, S. Y. Heo, and S. W. Lee, "Controlling human causal inference through in silico task design," *Cell Reports*, vol. 43, no. 2, p. 113702, 2024. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S2211124724000305

[12] L. J. Geneen, R. A. Moore, C. Clarke, D. Martin, L. A. Colvin, and B. H. Smith, "Physical activity and exercise for chronic pain in adults: an overview of cochrane reviews," *Cochrane Database of Systematic Reviews*, no. 4, 2017. [Online]. Available: https://doi.org//10.1002/14651858.CD011279.pub3

[13] C. Eccleston, E. Fisher, S. Liikkanen, T. Sarapohja, C. Stenfors, S. K. Jääskeläinen, A. S. Rice, L. Mattila, T. Blom, and J. R. Bratty, "A prospective, double-blind, pilot, randomized, controlled trial of an "embodied" virtual reality intervention for adults with low back pain," *PAIN*, vol. 163, no. 9, 2022. [Online]. Available: https://journals.lww.com/pain/fulltext/2022/09000/a_prospective,_double_blind,_pilot,_randomized,.10.aspx

[14] J. H. Shin, J. H. Lee, S. Tong, S. W. Kim, and S. W. Lee, "Designing model-based and model-free reinforcement learning tasks without human guidance," in *33rd Conference on Neural Information Processing Systems (NeurIPS 2019)*. Neural Information Processing Systems Foundation, 2019. [Online]. Available: http://hdl.handle.net/10203/270557

ACKNOWLEDGMENT

The authors would like to thank the members of the Institute of Biomedical Engineering and the Nuffield Department of Clinical Neurosciences at the University of Oxford for providing continuous feedback from their clinical and research experience, which was invaluable to the development of this project. We also extend our thanks to the attendants of the VR Pain workshop and other related meetings for their insightful comments and valuable suggestions on the software. Above all, we are deeply grateful to all those who contribute to chronic pain research as participants whose engagement with chronic pain research makes the development and clinical translation of platforms like this one possible.

CODE AVAILABILITY

https://github.com/ShuangyiTong/PainLabInteractiveControlPanel