TG2: A software package for behavioral neurophysiology and closed-loop spike train decoding

Daniel Morris
Stanford University
Department of Computer Science
dmorris@cs.stanford.edu

Abstract

Previous studies have demonstrated the feasibility of reconstructing intended arm movements from neural spike trains in real-time. Currently, research groups develop specialized environments for studying closed-loop control, and these environments typically depend on specialized hardware, multiple computers, and multiple software packages. This complexity limits the transfer of neural prosthetic work to a clinical environment and the rapid implementation of new decoding algorithms or paradigms in animal studies. In order to address these issues, we have implemented *TG2*, a Windows-based software package that integrates flexible behavioral control scripting with real-time neural decoding. A novel state-machine-based representation of experiments allows behavioral tasks to be developed graphically. Device-independent hardware abstractions allow experiments to be rapidly transferred among a variety of neural data acquisition systems, kinematic input devices, and neural decoding algorithms. An integrated simulation package allows offline development and debugging of real-time decoding algorithms. Although TG2 is designed for neurophysiology, it makes use of novel features that are applicable to a wide variety of behavioral experiments. We discuss the design and implementation of TG2, the novel software architecture and operator interface techniques that separate it from related software packages, and its initial use in both clinical and animal studies.

1. Introduction

An estimated 11,000 new patients each year suffer from partial or total paralysis as a result of spinal cord injury alone [1]. Consequently, the restoration of movement and/or communication in motor-impaired patients remains a major goal in neuroscience research. Although progress continues in the regeneration of damaged spinal cord tissue [2], several groups have recently demonstrated the potential of Brain-Computer Interface (BCI) technology as an alternative approach. Neural signals from motor cortex ([3],[4],[5]), parietal cortex [3], and other CNS areas have been successfully decoded in real time to allow direct cortical control of computer cursors or robotic actuators, bypassing the spinal cord entirely. These studies have spanned several species and numerous neural recording technologies.

Despite these successes, the engineering of real-time neural decoding systems remains a challenge that is often solved by lab-specific and even experiment-specific decoding systems. There are several systems available for stimulus generation and behavioral control (e.g. National Instruments LabView, PsychToolbox [6], and Reflective Computing's Tempo), but none of these systems provide integrated communication with the variety of neural data acquisition systems or kinematic input devices that are widely used in motor neurophysiology and/or real-time decoding experiments. As a result, research groups typically invest a significant amount of time building experimental platforms from several software packages, and it is often necessary to use several computers to manage the real-time decoding process.

This complexity limits both the rate at which experiments can be conducted and the ease with which decoding algorithms can be transferred among experiments or into a clinical environment. To address these issues, we have developed TG2, a software package that combines several aspects of real-time decoding – in addition to standard behavioral control and stimulus presentation – into a single software package. Several novel features separate TG2 from other packages currently in use for closed-loop neurophysiology studies; some of these features are summarized here:

Hardware Abstraction

TG2 provides device-independent abstractions for neural data and kinematic data that allow experiments to transfer easily among different environments. The experimental logic, stimulus presentation, and real-time analysis operate on device-independent coordinates and a device-independent representation of cells and spike trains. These device-independent representations are logged to TG2's data files (in addition to the raw input received from various devices), so offline analyses can also be transferred rapidly among a variety of experiments.

For example, for an ongoing experiment in which a primate subject is presented with a moving cursor, we are able to rapidly toggle between several sources of position information, including a joystick, a wrist-position tracker, and a neural decoding filter. Similarly, we are able to transfer the same experiment scripts among several sources of real-time spike data, including three neural recording systems and a simulation environment, without modifying our experiment script or offline analysis scripts.
Furthermore, rather than providing support for kinematic data acquisition from a small set of specific devices, TG2 provides communication with several general-purpose device interfaces, which are already implemented by numerous physical input devices. Examples include Microsoft's DirectInput interface and the WinTab interface for absolute pointing devices.

Graphical Programming

TG2 also allows users to define behavioral control scripts graphically, which greatly simplifies the process of creating experiments and allows researchers to develop complex experimental structures with limited programming time. This has the added benefit of allowing most experimental features to be optimized within the software package, so users do not have to be concerned about the efficient implementation of complex features or experimental logic.

External Interfaces

For those users that require features that are not available from the graphical interface, TG2 can communicate in real-time with Matlab or with custom C/C++ libraries, locally or over a LAN. Users can thus provide an interface between TG2 and custom experimental logic or custom hardware. This is also the mechanism by which users can develop custom decoding algorithms.

Integrated Simulation Environments

The debugging and testing of real-time decoding systems has also traditionally presented a challenge to research groups in this field. Real-time execution presents unique challenges associated with timing and performance, which reduces the effectiveness of offline debugging. With this in mind, TG2 was developed in parallel with two simulation environments that assist in the debugging of real-time algorithms and experiments. One of these environments "plays back" the spike train and kinematic stream collected from a previously recorded experiment, and the other uses a PC to generate spike trains based on input from a mouse or joystick.

Simulation is particularly essential for clinical experiments, when debugging time is limited or unavailable in the experimental environment.

Integrated Decoding Algorithms

In addition to allowing users to develop custom decoding algorithms, TG2 includes integrated, optimized implementations of several neural decoding algorithms that have been successfully employed by various research groups. Specifically, TG2 includes a linear filter [7], a Kalman filter [8], a spike rate integrator [9], and a maximum-likelihood estimator [10]. These integrated algorithms allow researchers to rapidly conduct experiments that don't require custom decoding algorithms; for example to explore user interfaces for BCI systems, training techniques for closed-loop control, or real-time control of novel actuators.

The remainder of this paper will describe the implementation of the above features and TG2's initial use in human and animal studies.

2. Methods

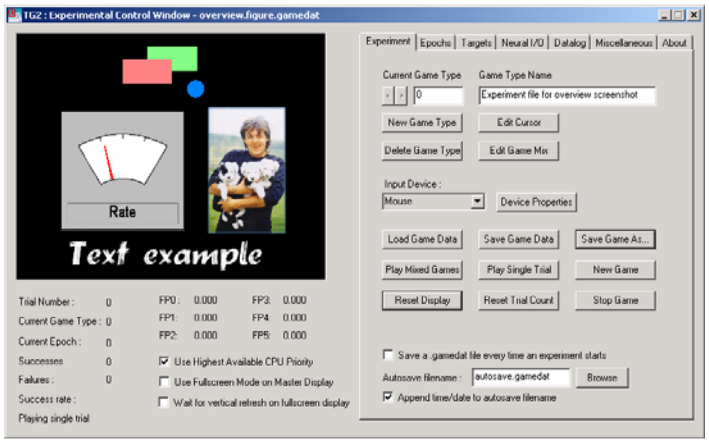
TG2 and supporting packages were developed in C++ (Microsoft Visual C++ 6.0). TG2 was tested on several desktop and laptop computers running Windows 2000 and Windows XP.

2.1 TG2 Description and Programming Model

TG2's core functionality provides behavioral task scripting and graphical display on two monitors (one for the subject and one for the operator). FIGURE 1 shows a screenshot of TG2; the task display area is a windowed version of the fullscreen display that is presented to the subject.

TG2's programming model is structured around four fundamental entities: "targets", "epochs", "events", and "conditions".

A "target" is a two-dimensional position and – optionally – its representation on one or two displays. FIGURE 1 shows several example target representations, including polygonal shapes, images, text, and "meters". Targets need not be displayed on the screen; a target's two-dimensional position can be treated as a two-point vector for computation, rather than the location of a visual object. Each target's position can be mapped to one of several input devices (see section 2.2), and a target can perform several transformations on the input it receives from the corresponding device. Available transformations include (but are not limited to) shifting, scaling, integration, differentiation, and low-pass filtering. One of the available "input devices" to which a target can be mapped is a pair of variables, or the position of another target. In this manner, targets – visible or invisible – can effectively be used as computational elements in a two-point data stream.



TG2 : Experimental Control Window - overview.figure.gamedat

Current Game Type: 0
Game Type Name: Experiment file for overview screenshot
Input Device : Mouse

Trial Number : 0
Current Game Type : 0
Current Epoch : 0
Successes 0
Failures : 0
Success rate :
Playing single trial

FP0 :	0.000	FP3:	0.000
FP1:	0.000	FP4:	0.000
FP2:	0.000	FP5:	0.000

[✓] Use Highest Available CPU Priority
[ ] Use Fullscreen Mode on Master Display
[ ] Wait for vertical refresh on fullscreen display

[ ] Save a .gamedat file every time an experiment starts
Autosave filename : autosave.gamedat
[✓] Append time/date to autosave filename

FIGURE 1: The main operator interface to TG2. The task area shows several example targets; these targets are (optionally) mirrored on the subject's full-screen display.

An "epoch" is a period of time in which TG2 is operating in a particular state; it is effectively one state of a user-defined finite-state machine. A user-defined set of "events" is executed at the beginning of each epoch, and an epoch ends when one or more of several user-defined "conditions" becomes true. Each condition in an epoch is associated with another epoch; when a particular condition becomes true, TG2 transitions to the specified epoch.

Example events include the manipulation of target visibility or position, the manipulation of digital output ports, the manipulation of internal variables (using arbitrary arithmetic expressions), the control of movie playback or target animation, sound playback, and speech synthesis.

Example conditions include timeouts, intersections among targets, value changes on digital input pins, input device events (e.g. button-presses), and arbitrary boolean expressions in terms of internal variables. Conditions can be "AND-ed" together to form more complex boolean expressions, and events can be executed either unconditionally or only if certain conditions are true when an epoch begins. Most events and conditions can also be parameterized in terms of internal variables that the user can control throughout an experiment. In this manner, arbitrarily complex experimental structures, making full use of optimized display features and complex hardware interaction features, can be built quickly from a reasonably small number of available events and conditions, without requiring low-level programming.

All epochs, events, conditions, and targets can be copied and pasted to minimize repetitive definitions. Additionally, an "asynchronous" list of conditions is constantly evaluated during any epoch. This allows users to define a single condition that should cause a transition regardless of the current state; for example, a trial might be aborted at any point if the subject releases an input device.

FIGURE 2 contains a pseudocode representation of the state machine used to execute an experiment, and FIGURE 3 shows the GUI used to edit the events and conditions associated with a particular epoch.

2.2 Input devices

Any target in TG2 can be mapped to an input device; the input from the device can optionally be transformed according to the functions discussed in section 2.1. Note that in addition to the functions described above, input device data can be transformed through arbitrary functions using TG2's arithmetic expression parser.

Since physical input devices tend to vary significantly across experiments and across research groups, the input device formats we chose to support in TG2 are intended to encapsulate a wide variety of possible physical input devices. Only one of the available input sources uses a manufacturer-specific API (SensAble Inc.'s "Ghost" API for Phantom haptic feedback devices [11]); each of the other TG2 "input devices" actually represents a set of physical devices that present a particular interface. In this manner, TG2 can be used with a variety of physical input devices, usually without requiring low-level programming to interface with a new device.

For example, a target can be mapped to any device that supports Microsoft's DirectInput standard. This includes commercial game devices (joysticks, gamepads, etc.) and a variety of other devices. A target's position can be mapped to any combination of the numerous axes defined by the DirectInput standard. This approach has been used to interface TG2 with several input devices, including several standard desktop joysticks and several non-ferromagnetic joysticks designed for use in fMRI environments (Resonance Technology).

Similarly, a target can be mapped to any device that supports the WinTab (LCS/Telegraphics) API, a standard for interfacing with absolute pointing devices. We have used this approach to collect kinematic training data for neural filtering algorithms, in humans (intraoperatively) and non-human primates (see results in section 3).

The Windows system cursor serves as another general-purpose input device; a target can be mapped to a transformed version of the Windows cursor position, which allows TG2 to interface with devices that do not provide a low-level API but are able to manipulate the Windows cursor. We have used this approach to interface with a touchscreen (Keytec, Inc.) and with the mouse, which we use for demonstrations and pre-operative patient training.

```
curEpoch := 0;

while(1) {
  execute all events associated with epoch curEpoch;
  while(1) {
    for each condition associated with epoch curEpoch {
      if (the current condition is true) {
        if (the current condition ends the experiment) exit;
        curEpoch := (the epoch associated with the current condition);
        break;
      }
    }
    if (any conditions in this epoch became true) break;
  }
}
```

FIGURE 2: A pseudocode representation of TG2's finite state machine.

An additional input stream, which we refer to as a "library input device", allows the user to specify a Windows Dynamic Link Library (DLL) that contains a custom interface to an unsupported data stream. The TG2 package includes a header file that describes the interface TG2 expects to use to communicate with library input devices. Users whose hardware does not conform to any supported standards can implement the specified functions in a Windows DLL to connect TG2 to custom hardware. This interface can also be used for writing custom decoding algorithms; TG2 treats library devices as an arbitrary source of input data and can provide arbitrary experiment state information to the library. We have used this approach to support a wrist-position-tracking device that transmits data to a digital input card; this device is currently being used in primate physiology experiments.

In addition to the "input devices" supported by TG2 – which represent continuous streams of position or numeric information – TG2 can also process discrete events from the keyboard or from digital input sources. The "digital input" condition allows TG2 to transition among epochs based on the state of a digital input port; native support is provided for the parallel port and any Advantech digital input card. We have used this approach to connect TG2 to a hand-switch which our primate subjects are required to hold during unilateral arm movement experiments.

2.3 Neural data processing

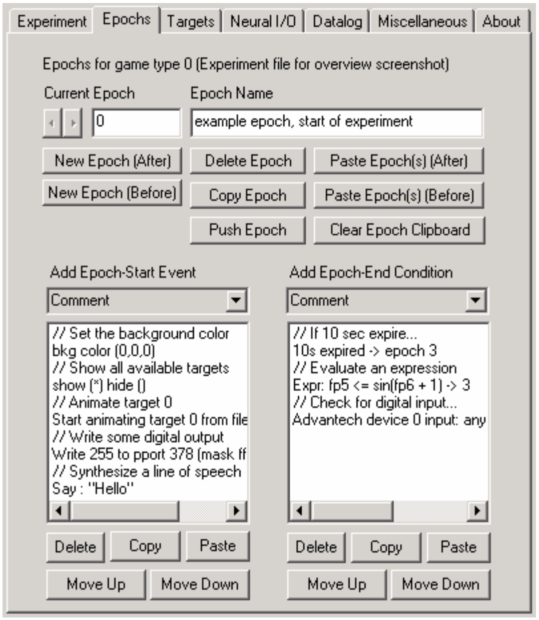
Several unique problems arise in real-time communication with neural data acquisition devices. There is no standard format for network transmission or file storage of neural data (although the NeuroShare project (http://neuroshare.org) represents progress toward a common interface for file access). This non-uniformity requires researchers using multiple systems to modify online and offline analyses to deal with the specific formats offered by particular manufacturers. Additionally, real-time aspects of decoding software generally have to be debugged and tested on each system, to account for varying data formats and timing characteristics.

In order to simplify online and offline analyses, TG2 has native support for communicating with several data acquisition systems (including systems from Plexon, Cyberkinetics, and Alpha-Omega Engineering), and allows the user to operate on a device-independent representation of neural data. The decoding algorithms incorporated natively into TG2 (see section 2.4) are parameterized in terms of device-independent "channels" and "units", and the data logged to TG2's data files is in a common format that's independent of the recording system (although system-specific configuration information is also logged to disk).

An additional challenge in real-time decoding arises from the fact that training stimuli and kinematics are generated/acquired on a clock that is independent of the clock used on the recording hardware. In order to synchronize the data necessary for online reconstruction, researchers are typically required to generate device-specific synchronization codes. TG2 handles this synchronization internally, by generating periodic digital pulses on a digital output card (the parallel port and Advantech PCI digital output cards are supported). The supported neural recording systems respond to digital codes with timestamped network packets; TG2 extracts synchronization information from the timestamped responses. In this manner, all neural data can be written to TG2's data files or made accessible online with timestamps that are synchronized to the behavioral stimulus clock, and users do not have to be concerned with synchronization adjustments (device timestamps are also recorded to disk and made available online).

2.4 Network interface

In addition to providing access to TG2 state information via the "library device" interface, which requires the development of a custom C/C++ library, TG2 also includes a TCP/IP server that allows clients to access and manipulate TG2 state variables. Via this interface, clients can modify and access internal



Epochs for game type 0 (Experiment file for overview screenshot)

Current Epoch: 0

Epoch Name: example epoch, start of experiment

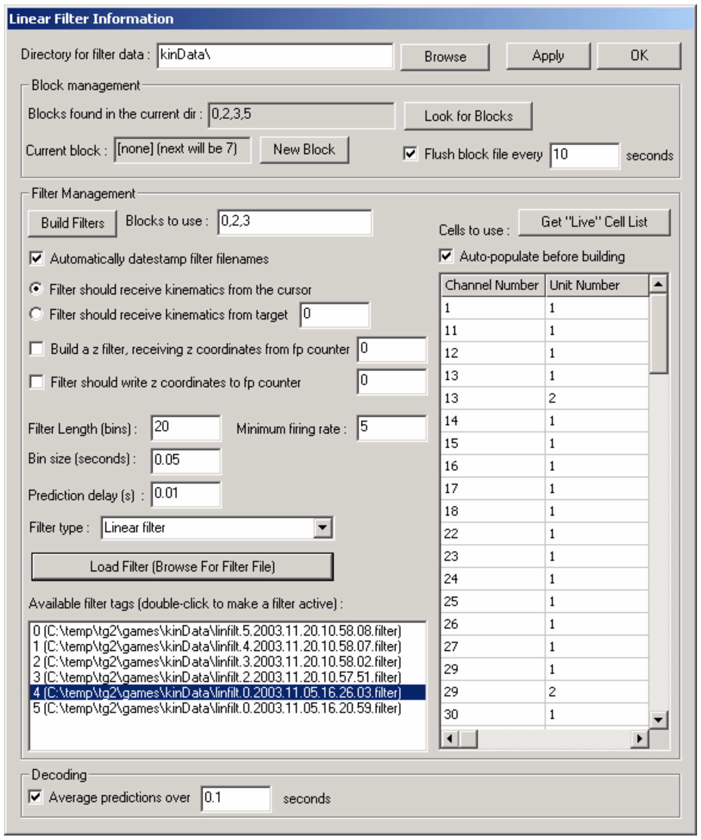
Add Epoch-Start Event: Comment

```
// Set the background color
bkg color (0,0,0)
// Show all available targets
show (*) hide ()
// Animate target 0
Start animating target 0 from file
// Write some digital output
Write 255 to pport 378 (mask ff
// Synthesize a line of speech
Say : "Hello"
```

Add Epoch-End Condition: Comment

```
// If 10 sec expire...
10s expired -> epoch 3
// Evaluate an expression
Expr: fp5 <= sin(fp6 + 1) -> 3
// Check for digital input...
Advantech device 0 input: any
```

FIGURE 3: An example of TG2's GUI-based programming environment. The interface for editing a specific epoch is shown. Several events are defined on the left; these events are executed when this epoch begins. Several conditions are defined on the right; when one of these conditions becomes true during an experiment, TG2's state machine jumps to the corresponding epoch.



Linear Filter Information

Directory for filter data : kinData\

Block management

Blocks found in the current dir : 0,2,3,5

Current block : [none] (next will be 7)

☑ Flush block file every 10 seconds

Filter Management

Blocks to use : 0,2,3

Cells to use :

☑ Auto-populate before building

☑ Automatically datestamp filter filenames

● Filter should receive kinematics from the cursor

○ Filter should receive kinematics from target 0

☐ Build a z filter, receiving z coordinates from fp counter 0

☐ Filter should write z coordinates to fp counter 0

Filter Length (bins) : 20

Minimum firing rate : 5

Bin size (seconds) : 0.05

Prediction delay (s) : 0.01

Filter type : Linear filter

Available filter tags (double-click to make a filter active) :

```
0 (C:\temp\tg2\games\kinData\linfilt.5.2003.11.20.10.58.08.filter)
1 (C:\temp\tg2\games\kinData\linfilt.4.2003.11.20.10.58.07.filter)
2 (C:\temp\tg2\games\kinData\linfilt.3.2003.11.20.10.58.02.filter)
3 (C:\temp\tg2\games\kinData\linfilt.2.2003.11.20.10.57.51.filter)
4 (C:\temp\tg2\games\kinData\linfilt.0.2003.11.05.16.26.03.filter)
5 (C:\temp\tg2\games\kinData\linfilt.0.2003.11.05.16.20.59.filter)
```

Channel Number	Unit Number
1	1
11	1
12	1
13	1
13	2
14	1
15	1
16	1
17	1
18	1
22	1
23	1
24	1
25	1
26	1
27	1
29	1
29	2
30	1

Decoding

☑ Average predictions over 0.1 seconds

FIGURE 4: TG2's user interface for managing linear and Kalman decoding filters.

numeric variables and target positions; clients can also start and stop experiments. To avoid request/response latencies, clients can request that a particular set of target positions or variable values be periodically streamed over the network.

Users can access the network server from arbitrary client software (the protocol and packet format are made available to researchers using TG2). We have also provided a C++ library that manages the low-level connection state and a Matlab wrapper for this library. Thus, users can access and manipulate experimental state information in real-time from Matlab, either on the machine on which TG2 is running or on a remote machine. This is the recommended approach for researchers developing complex behavioral logic that is not available from the GUI or users developing custom neural decoding filters. This approach allows users to utilize the graphics, data logging, and hardware interface features available within TG2, while still developing completely custom behavioral tasks from a fully scripted language.

Additionally, we have developed several clients that use the TG2 network server for a variety of external applications. These clients serve as both a demonstration of the network server's functionality and a set of extensions to the core TG2 functionality:

- *TG2-3D*: An increasing number of projects in motor physiology and neural prosthetics require three-dimensional movements and three-dimensional stimulus display. Since the native display system in TG2 represents only two-dimensional objects, we have developed an external client that receives state information and target position information from TG2 and renders targets moving in three dimensions. This can be run on the same computer as TG2 itself, but it offers the option of offloading time-consuming rendering operations onto a separate computer.

  FIGURE 5 shows an example of a set of targets displayed in TG2 and in the TG2-3D client. The client is shown here in windowed mode, but it can also be run in fullscreen mode and in page-flipped stereo fullscreen mode, for presentation of stereo images to a subject.

- *Desktop Control Client*: As a demonstration of TG2's application to brain-computer interfaces, and as a tool for users interested in exploring user interface issues associated with BCI's, we have also developed a client that connects to TG2's network server and synthesizes mouse and keyboard events on a remote desktop based on filter predictions generated in TG2. The client can also synthesize speech from the keystrokes supplied by TG2 (using the Microsoft Speech API). FIGURE 6 shows the user interface for the "desktop control client".

2.5 Integrated decoding algorithms

Many research applications do not require custom decoding algorithms. For example, users interested in exploring user interface aspects of brain-computer interfaces or the control of novel robotic actuators from neural signals may prefer to use optimized implementations of previously-developed decoding algorithms.

With this in mind, TG2 includes native implementations of several neural decoding algorithms; each allows graphical configuration of relevant parameters to adjust decoding performance for specific experiments. The user configures these parameters in a filter-specific GUI, and specifies a target or set of "targets" (see section 2.1) to use as a source of kinematic training data. Values computed by the filters can be written to internal variables – accessible via events and conditions (see section 2.1) – or directly to target positions.

The four algorithms that are included in TG2 are described briefly here, along with information about their integration into TG2.

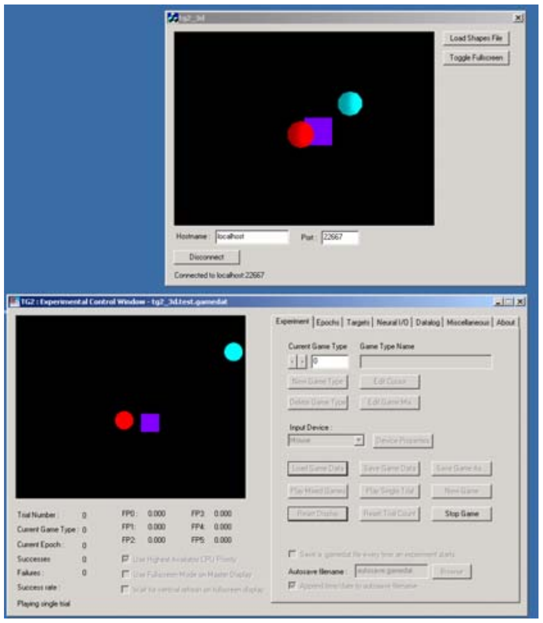
- An optimal linear filter, as presented in [7] and employed in closed-loop primate decoding in [4].

  The parameters used to define the linear filter – specifically the filter length and bin size – can be controlled from a GUI (see FIGURE 4). Additionally, the subset of active cells to be used for decoding is controllable from the GUI. We have found that a practical approach to rapidly selecting a meaningful subset of available cells is to sample only those cells that fall above a specified mean firing rate over the training period. This method of selecting cells is available from the linear filter GUI, which allows the user to avoid manually selecting individual cells.

  A list of filters that have been generated is available in the GUI with descriptive information; a user can rapidly toggle among available filters to evaluate their online performance. Filters are also saved to disk and can be re-loaded in future sessions. Furthermore, training data can be collected in data "blocks" of arbitrary length, and filters can be generated from arbitrary sets of data blocks. This allows a user to experiment with various combinations of training data periods and to eliminate periods of low-quality behavioral data from filter training.

  The native linear filter allows regression onto two or three kinematic signals (typically x, y, and z hand position). Results obtained using the native linear filter implementation in primate experiments are presented in section 3.

- A Kalman filter, as presented in [8]. The parameters used to configure the Kalman filter are similar to those used to configure the linear filter, so a common GUI (FIGURE 4) is used to control both filters.



Hostname : localhost Port : 22667

Connected to localhost:22667

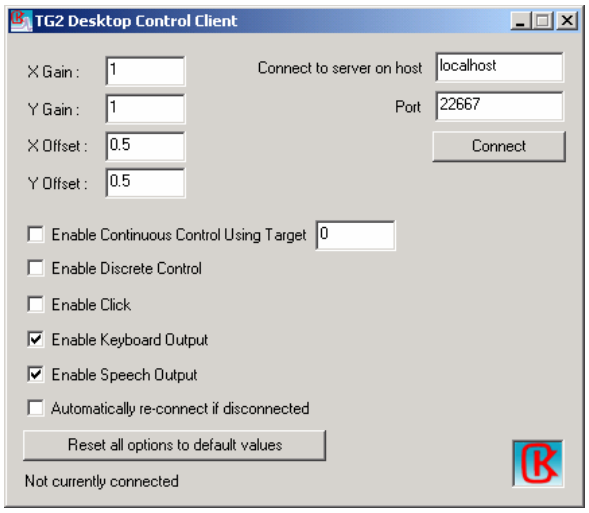
Trial Number :	0	FP0 :	0.000	FP3 :	0.000
Current Game Type :	0	FP1 :	0.000	FP4 :	0.000
Current Epoch :	0	FP2 :	0.000	FP5 :	0.000
Successes :	0				
Failures :	0				
Success rate :					

Playing single trial

Current Game Type: 0
Game Type Name:
Input Device : Mouse

[✓] Use Highest Available CPU Priority
[ ] Use Fullscreen Mode on Master Display
[ ] Wait for vertical refresh on fullscreen display
[ ] Save a gamedat file every time an experiment starts
Autosave filename : autosave.gamedat
[✓] Append time/date to autosave filename

FIGURE 5: The TG2-3D client, displaying a 3d rendering (top) of the 2d targets displayed in TG2's main console (bottom). Depth values are drawn from TG2 state variables that the user associates with each target.


TG2 Desktop Control Client
Connect
Reset all options to default values

X Gain : 1
Y Gain : 1
X Offset : 0.5
Y Offset : 0.5
Connect to server on host: localhost
Port: 22667

[ ] Enable Continuous Control Using Target 0
[ ] Enable Discrete Control
[ ] Enable Click
[✓] Enable Keyboard Output
[✓] Enable Speech Output
[ ] Automatically re-connect if disconnected

Not currently connected

Figure 6: The user interface for the TG2 "Desktop Control Client".

- A maximum-likelihood estimator (MLE), as presented in [10]. The MLE is trained to classify discrete states based on the observed mean firing rates of each sampled unit around the behavioral state. Training states are specified via "discrete classifier state" events. For example, in a multiple-direction center-out task, the user would typically include a "discrete classifier state" event around the time of directional stimulus.

The user can graphically configure the time window around a state that's used for classification, along with the subset of available cells that are used for classification. Estimators can be saved to disk and recalled in later session.

- A "leaky" rate integrator, as presented in [9]. This is one of the simplest approaches to real-time decoding; a continuous value is computed as a weighted sum of the firing rates of a number of cells. The user can graphically adjust the subset of cells being sampled, the time constant for rate integration, and the minimum and maximum output values.

2.6 Data logging

The abstraction of hardware devices is a major advantage of TG2's architecture, particularly because data files can be recorded in a device-independent format and can thus be transferred easily among offline analysis programs.

TG2 (optionally) generates four data files per experiment; each is a simple fixed-format, tab-delimited ASCII file that is parsed easily by offline analysis programs (e.g. Matlab or Excel). The time and date at which an experiment was started is (optionally) automatically incorporated into the filename of each data file; this greatly simplifies offline indexing and analysis.

- A kinematic data file includes device positions received from various input devices, both in native device coordinates and in TG2's device-independent coordinate system. Similarly, device-specific timestamps are recorded to the data log along with device-independent "game time" values, to allow simple synchronization among multiple devices.

- A spike data file includes a timestamp for each spike delivered by the neural recording hardware, both as a device-independent TG2 "game time" and as a timestamp on the clock used by the recording hardware.

- An "event" data file contains a representation of all events that were executed during an experiment, and all state transitions ("conditions" that evaluated to "true").

- A "filter information" data file contains a representation of all neural filters that were built, loaded, or modified during an experiment.

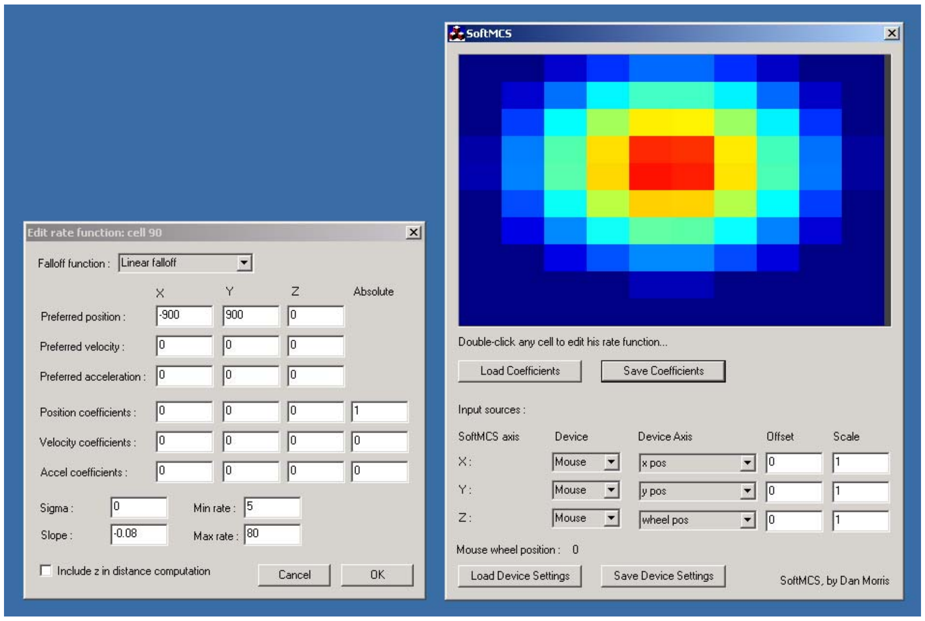
Because these files are in a common format and use a common clock, it is easy for a user to re-construct an entire experiment offline for analysis, debugging, or presentation.

Since it is not practical to capture all neural waveform information in TG2's data files when they are already being collected by the recording hardware, TG2 also includes a facility for sending behavioral codes to neural recording hardware for offline synchronization. The user can use a "digital output event" to write data to a particular hardware port, and TG2 will handle the low-level details of handshaking with the recording hardware. Similarly, TG2 can be configured to send a kinematic (target position) stream to a neural recording system as a series of digital codes. Thus users that prefer to use the neural recording system's data files online or offline can have access to the kinematic and behavioral events generated by TG2 via the recording system.

2.7 Simulation

Real-time decoding algorithms and applications are difficult to design and test offline; problems related to event timing and computational efficiency often arise only when algorithms are run in real-time. Furthermore, for clinical applications, it is often helpful to be able to "practice" experimental paradigms outside of the clinical environment. With this in mind, we have developed two simulation packages that send neural data to TG2 in real-time over a network, emulating the behavior of a neural recording system that is recording from a subject.

The first simulation package is called "SoftMCS" (for "software motor cortex simulator"); this software takes input from a mouse or joystick and – via a set of user-defined functions that govern the firing rates of each simulated cell – generates a spike train that reflects the position, velocity, and/or acceleration of each input device axis. The user defines coefficients that transform device position into a firing rate for each cell, and this rate is used to drive a Poisson process that generates the actual spiking pattern. Spikes are sent over the network in a format that mimics the output of the Cerebus Data Acquisition System (Cyberkinetics, Inc.). FIGURE 7 shows a screenshot of the SoftMCS program.



Edit rate function: cell 90

Falloff function : Linear falloff

	X	Y	Z	Absolute
Preferred position :	-900	900	0	
Preferred velocity :	0	0	0	
Preferred acceleration :	0	0	0	
Position coefficients :	0	0	0	1
Velocity coefficients :	0	0	0	0
Accel coefficients :	0	0	0	0

Sigma : 0 Min rate : 5
Slope : -0.08 Max rate : 80

☐ Include z in distance computation

Input sources :

SoftMCS axis	Device	Device Axis	Offset	Scale
X :	Mouse	x pos	0	1
Y :	Mouse	y pos	0	1
Z :	Mouse	wheel pos	0	1

Mouse wheel position : 0

FIGURE 7: A screenshot of the "SoftMCS" program. The window on the left allows the user to modify the function that drives a simulated cell (these functions can also be loaded in batch from a file). The window on the right shows the current firing rate of 100 simulated cells and allows the user to modify the kinematic input sources.

The second simulation package we have developed as a complement to TG2 reads previously-recorded data files (either TG2 data files or data recorded in Plexon's .plx file format) and streams them back to TG2 in real-time, simulating the output of a Plexon or Alpha-Omega neural recording system. If kinematic and neural information are available in the selected data files, an entire experiment can be repeated offline, in simulated real-time.

3. Results

We will briefly discuss our experiences with the application of TG2 to various experimental environments and various neural recording systems. Quantitative results will be presented in some cases, but length constraints prevent us from describing all experiments in detail.

3.1 Closed-loop primate studies

A female rhesus monkey was trained to move a cursor in two dimensions to a series of randomly presented targets. Kinematic data was acquired using a Wacom Intuos2 graphics tablet. During movement, neural data was acquired using a Cerebus Data Acquisition System (Cyberkinetics, Inc.). Kinematic and spike data were used to train a linear filter (see section 2.5), and for several minutes in each session, cursor position was driven by the output of TG2's linear filter.

FIGURE 8 shows the relationship between hand position and filter prediction during one "neural control" period. The subject continued to successfully acquire targets during this period. Predicted hand position correlated to actual hand position with a mean absolute error of 4.7 cm.

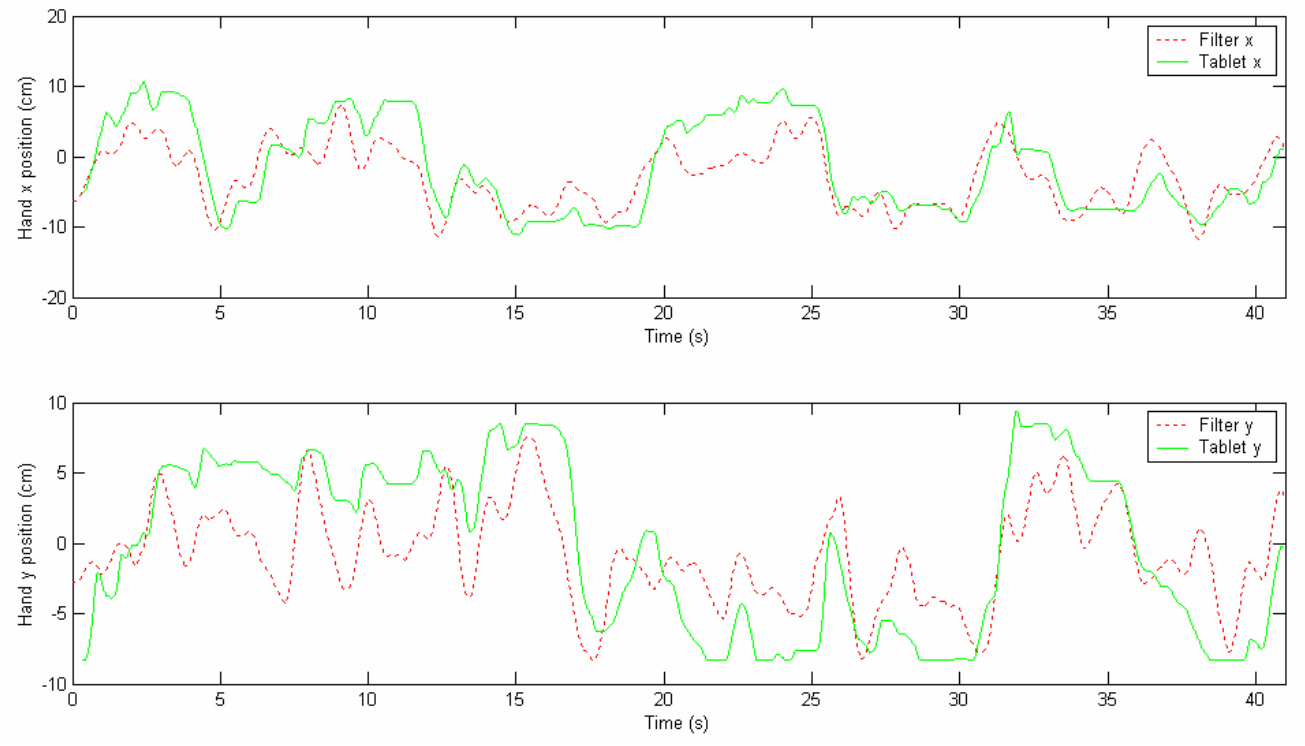


FIGURE 8: TG2 hand position predictions vs. actual hand position for one primate experiment. The top plot shows the predicted and actual x positions, the bottom plot shows y positions.

3.2 Clinical studies

TG2 is currently being used in a set of clinical studies in which spike data is collected intraoperatively from human motor and premotor cortex (via an Alpha-Omega data acquisition system) and transferred to TG2 in real-time. Subjects are asked to perform a center-out task, and kinematic data is collected via a Wacom digitizing tablet. TG2 is used to build decoding filters (linear filters and maximum likelihood estimators), which drive a stimulus that is displayed on the subject's monitor.

TG2 enabled the development and testing of the relevant experimental scripts and decoding filters to take place offline or in simulation, with rapid transfer to the clinical environment. Furthermore, TG2 enabled the use of nearly-identical behavioral tasks in preoperative functional imaging studies, allowing assessments of task-related neural response via fMRI and electrophysiology.

Preliminary clinical decoding results were presented in [12].

3.4 Incorporation into a novel primate experiment

TG2 allowed the rapid incorporation of a novel hardware device constructed to measure wrist movements in humans and monkeys. The intention is to allow a subject to perform a target-acquisition task using both a two-dimensional hand tracker (a graphics tablet) and the wrist tracker, with identical task structure and visual display.

A simple dynamic-link library (adhering to the "library input device" format) was developed to interface with the device, and all other aspects of the experiment – including experimental logic, target presentation, reward timing, etc. – worked as they had with the hand-tracking task.

4. Conclusion

The results described above demonstrate that TG2 allows experimenters to easily design a wide variety of behavioral tasks that interact with a wide variety of hardware devices. Furthermore, TG2 allows experiments to transfer rapidly among experimental environments, which should greatly ease the integration of primate results into clinical studies.

We propose that as neurophysiology becomes increasingly focused on translational research, novel abstractions like those provided in TG2 will be incorporated into other software packages to similarly accelerate the transition from animal to human experiments.

References

[1] A.I. Nobunago, B.K. Go, RB. Karunas, Recent demographic and injury trends in people served by the Model Spinal Cord Injury Care System, Arch Phys Med Rehabil, 1999; 80: 1372-82.
[2] J.W. McDonald and D. Becker, Spinal cord injury: promising interventions and realistic goals, Am J Phys Med Rehabil, 2003; 82(10 Suppl):S38-49.
[3] J. Wessberg, C.R. Stambaugh, J.D. Kralik, P.D. Beck, M. Laubach, J.K. Chapin, J. Kim, S.J. Biggs, M.A. Srinivasan, M.A.L. Nicolelis, Real-time prediction of hand trajectory by ensembles of cortical neurons in primates. Nature, 2000; 408: 361-365.
[4] M.D. Serruya, N.G. Hatsopoulos, L. Paninski, M.R. Fellows, J.P. Donoghue, Instant neural control of a movement signal, Nature, 2002; 416(6877): 141-2.
[5] D.M. Taylor, S.I. Tillery, A.B. Schwartz, Direct cortical control of 3D neuroprosthetic devices, Science, 2002; 296(5574):1829-32.
[6] DH. Brainard, The Psychophysics Toolbox, Spatial Vision, 1997; 10: 433-436.
[7] D.K. Warland, P. Reinagel, M. Meister, Decoding visual information from a population of retinal ganglion cells, J Neurophysiol, 1997; 78(5): 2336-50.
[8] W. Wu, M.J. Black, Y. Gao, E. Bienenstock, M. Serruya, J.P. Donoghue, Inferring hand motion from multi-cell recordings in motor cortex using a Kalman filter, SAB'02-Workshop on Motor Control in Humans and Robots: On the Interplay of Real Brains and Artificial Devices, 2002; 66-73.
[9] E.E. Fetz, M.A. Baker, Operantly conditioned patterns on precentral unit activity and correlated responses in adjacent cells and contralateral muscles, J Neurophysiol, 1973; 36: 179-204.
[10] P. Földiàk, The “ideal homunculus”: statistical inference from neural population responses, Computation and Neural Systems, Eeckman F and Bower J, editors, Kluwer Academic Publishers: Norwell, MA, 1993; 55-60.
[11] T.H. Massie and J.K. Salisbury, The PHANTOM Haptic Interface: A Device for Probing Virtual Objects, Proceedings of the ASME Winter Annual Meeting, Symposium on Haptic Interfaces for Virtual Environment and Teleoperator Systems, 1994.
[12] J.P. Donoghue, M. Saleh, A. Caplan, D.S. Morris, S. Ramchandani, C. Ojakangas, G. Friehs, Direct Control Of a Computer Cursor by Frontal Cortical Ensembles in Humans: Prospects for Neural Prosthetic Control, Abstract presented at the Society For Neuroscience Annual Meeting, 2003; Program No. 607.9.