Journal of Neuroscience Methods 142 (2005) 27–34

JOURNAL OF NEUROSCIENCE METHODS

www.elsevier.com/locate/jneumeth

# A software solution for the control of visual behavioral experimentation

Travis Meyer, Christos Constantinidis*

*Department of Neurobiology and Anatomy, School of Medicine, Wake Forest University, Medical Center Blvd., Winston-Salem, NC 27157-1010, USA*

Received 5 May 2004; received in revised form 13 July 2004; accepted 16 July 2004

**Abstract**

Psychophysical and neurophysiological research requires precise control of experimental devices for the purpose of delivering stimuli and monitoring behavioral and neural responses. This has previously been accomplished by complex, often proprietary, programmable systems, interfacing with a limited range of hardware. We have developed a software solution entirely within the Matlab environment that can achieve high-speed control of experimental and behavioral variables. We make this Wake-Forest Visual Experimentation (WaVE) software freely available under the GNU public license, and demonstrate how to customize it to individual laboratory needs. WaVE takes advantage of existing Matlab libraries and toolboxes to present visual stimuli, collect experimental data, update behavioral variables, and communicate with other computers. Although we have developed it for use in a Windows-based Personal Computer, the portability of the Matlab code makes possible its customization for use in a variety of other systems. We present simulation results showing sub-millisecond sampling rate and updating precision, running on single-processor, desktop PCs. The WaVE software offers a simple, flexible and powerful solution that compares favorably with many of its costly alternatives.
© 2004 Elsevier B.V. All rights reserved.

*Keywords:* Psychophysics; Neurophysiology; Instrumentation; Software; Behavioral control; Vision; Eye movement; Saccade

## 1. Introduction

Behavioral neuroscience has benefited enormously by the development of computer hardware and software that made possible the control and automation of experiments. Computer interfaces can precisely time the delivery of sensory stimulation, the responses of a subject, and the concomitant neural signals. Such control allows for the design of complex behavioral paradigms and the ability to correlate neural activity with specific stimulus attributes and response components. Ever more sophisticated computer programs have been designed to harness the power of modern computers and provide correspondingly more advanced capabilities.

While there is no scarcity of software packages for behavioral experimentation, existing systems are either proprietary and costly, hardware-specific and therefore limiting, or enormously complex, requiring the mastery of their own scripting languages. We sought to design an inexpensive and simple, yet powerful alternative. Our attempt was motivated by the practical need of using such a system for our own neurophysiological investigation in a newly established laboratory. We wished for the software to run on relatively low-end personal computers and to be easy to use, without compromising its power. The minimum requirements of a software solution that could fit this description were:

(i) Ease of programming and customization. We opted to develop the software in the Matlab computer environment (Mathworks, Inc.). The Matlab suite is mature, fairly easy to use, and provides advantages, such as a programmable graphical user interface. Additionaly, Matlab is becoming the de facto standard in the analysis of neural and behavioral results. A large community of experimenters is already familiar with its use, and

* Corresponding author. Tel.: +1 336 716 7424; fax: +1 336 716 4534.
*E-mail address:* cconstan@wfubmc.edu (C. Constantinidis).

0165-0270/$ – see front matter © 2004 Elsevier B.V. All rights reserved.
doi:10.1016/j.jneumeth.2004.07.009

mastering its abilities by new students and research personnel is a worthwhile investment.

(ii) Ability to display complex visual stimuli. Precise control of the visual display was achieved through the Psychophysics Toolbox (Brainard, 1997; Pelli, 1997). This is an open-source suite of Matlab functions with advanced display capabilities.

(iii) Real-time feedback to the investigator on the progress of each behavioral trial. We used the native Matlab graphing abilities to display behavioral variables, such as the subject's eye position and stimulus location, on-line, with minimal programming effort, avoiding the implementation of an extensive graphical subsystem for this purpose.

(iv) Ability to sample inputs and direct outputs to hardware devices with sub-millisecond resolution. We controlled a variety of hardware devices directly through Matlab using the MathWorks Data Acquisition Toolbox and an input/output computer board. Real-time sampling resolution of such devices has traditionally been a problem for Windows and Macintosh computers, steering some developers to use specialized operating systems, which however narrowed the options of hardware these systems could be interfaced with and only added to the complexity and intricacy of the software. We found that the computing power of the last generation of PC processors has overcome the limitations of the operating system. Matlab, running on a single-processor Windows PC could reliably sample multiple hardware devices at sub-millisecond rates.

(v) Ability to receive and transmit messages to other computers on-line. We achieved that goal using the TCP/UDP/IP toolbox, an open-source library of network functions. Such a capability is necessary for the most complex applications; although a data acquisition computer board can directly sample behavioral parameters, such as eye position and button presses, more complex data, e.g. neurophysiological recordings from multiple microelectrodes, may necessitate specialized hardware and software. We were able to interface the computer running our visual experimentation software with a second computer dedicated to neurophysiological data collection through the functions of the TCP/UDP/IP toolbox. The same functionality could be provided by the MathWorks Instrument Control Toolbox.

The result of our effort, the Wake-Forest Visual Experimentation (WaVE) software, can be easily customized for use in psychophysical, fMRI and neurophysiological laboratories. We present simulation results documenting its performance and we make the software freely available under the GNU general public license (http://www.gnu.org/licenses/gpl.txt). The source code, together with instruction pages and examples can be obtained through an e-mail request to the authors.

## 2. Material and methods

### 2.1. Hardware and software requirements

The full WaVE software requires a PC connected to two or three monitors, a Data Acquisition Board, Matlab, and three toolboxes: the MathWorks Data Acquisition Toolbox, the Psychophysics Toolbox and the TCP/UDP/IP Toolbox. The cost of the system (excluding the subject monitor, the price of which can vary greatly depending on the particular needs of each laboratory) was approximately US$ 4000. We also make available a demonstration version of the software which only requires Matlab and the Psychophysics Toolbox, and can be easily tested in typical desktop systems. The demonstration program implements a delayed-response task, a visually-guided saccade and a fixation-only task, using the mouse cursor to simulate the subject's eye position.

#### 2.1.1. Computer configuration

We used a Dell Dimension 8300 Personal Computer produced in October 2003 to develop and evaluate the performance of the software. The computer was equipped with a 3.0 GHz Pentium 4 processor of 800 MHz bus speed, 512 Mb of RAM, and an integrated Intel PRO 10/100 Ethernet network board. The system included a 128 Mb Radeon 9800 Pro graphics card (ATI systems), with a dual monitor capability and a resolution of up to 1280 by 1024 pixels. We further split one of the monitor signals into two identical displays, so that the display presented to the subject could also be viewed by the experimenter, by using a two-video splitter and amplifier (Iogear). In addition, we installed a 12-bit, Input/Output Data Acquisition, PCI board: PCI-MIO-16E-4 (now available as PCI-6040E, National Instruments). We refer to this system as the Behavioral Control Computer (Fig. 1).

#### 2.1.2. Required software

The Behavioral Control Computer ran the Windows XP (Microsoft) operating system. We installed Matlab version 6.5, release 13, as well as the MathWorks Data Acquisition Toolbox version 2.2, which was used to control the Data Acquisition Card. We additionally installed the Psychophysics toolbox version 2.50, an open source function library providing a set of functions for the presentation of visual stimuli (http://psychtoolbox.org/). Finally we installed the TCP/UDP/IP toolbox version 2.0.5, an open-source library of network functions (http://www.mathworks.com/matlabcentral/fileexchange/loadFile.do?objectId=345&objectType=file).

#### 2.1.3. Additional hardware for experimentation and neural data acquisition

We incorporated a number of hardware devices in our experimental setup (Fig. 1). None of these are necessary for the WaVE software to operate but they are typical of the range of hardware used by visual research laboratories. The descrip-

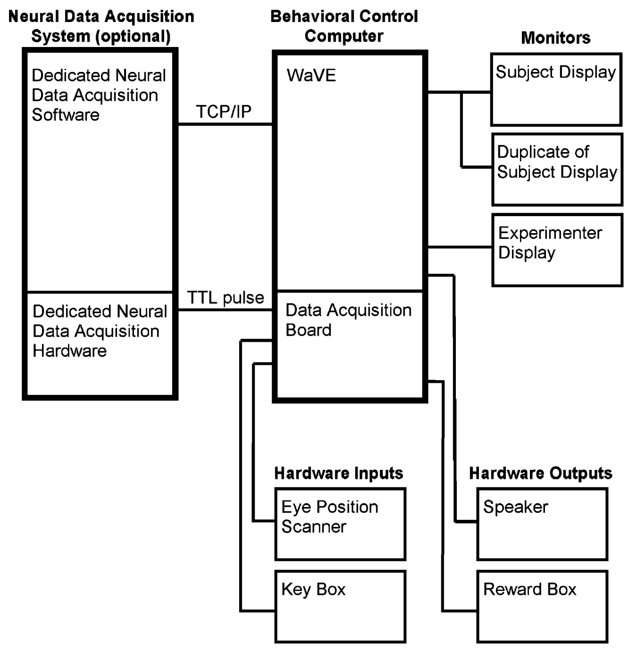

Neural Data Acquisition System (optional)
Dedicated Neural Data Acquisition Software
Dedicated Neural Data Acquisition Hardware
TCP/IP
TTL pulse
Behavioral Control Computer
WaVE
Data Acquisition Board
Monitors
Subject Display
Duplicate of Subject Display
Experimenter Display
Hardware Inputs
Eye Position Scanner
Key Box
Hardware Outputs
Speaker
Reward Box

Fig. 1. Schematic diagram illustrating the experimental setup. A personal computer running the WaVE software is shown in the center (behavioral control computer). This computer uses a data acquisition board to communicate with a number of hardware input and output devices (bottom). A second personal computer in our setup was dedicated to neurophysiological data acquisition (left).

tion of how they were interfaced with the Behavioral Control Computer is therefore instructive of how other laboratory setups can be customized. An infra-red, eye-position monitor (ISCAN) provided on-line eye position of our subjects (monkeys). A custom-made key box was used to input responses from the subject and generated a pulse when pressed. The Behavioral Control Computer generated outputs through a speaker for audio feedback, and through a custom-made device which delivered a liquid reward when triggered by an electrical pulse at the end of a behavioral trial. Other devices that could be easily accessed through WaVE include the keyboard and mouse, or any hardware that could be connected to the keyboard, mouse or USB port.

A second computer and Data Acquisition system (APM system, FHC Inc.) was dedicated to the collection of neurophysiological data from 16 electrode channels. The Behavioral Control Computer was interfaced with the Neural Data Acquisition system in two ways: synchronization of behavioral and neural data was achieved through a digital pulse generated by WaVE and transmitted to the Neural Data Acquisition system through a custom-made interface cable. Additionally, the two computers were linked through an Ethernet Network. A second Matlab session running on the Neural Data Acquisition system received input from the WaVE software through TCP/IP functions. In this way we were able to transmit complex variables to the Neural Data Acquisition system, such as a description of the size, color and shape of the visual stimulus being presented. A number of other Neural Data Acquisition systems (e.g. by Plexon, CED Electronics, Alpha-Omega, Neuralynx) offer equivalent data acquisition capabilities and could be interfaced with WaVE in a similar manner.

## 2.2. *Implementation of a delayed-saccade task for monkey neurophysiology*

### 2.2.1. *Behavioral task*

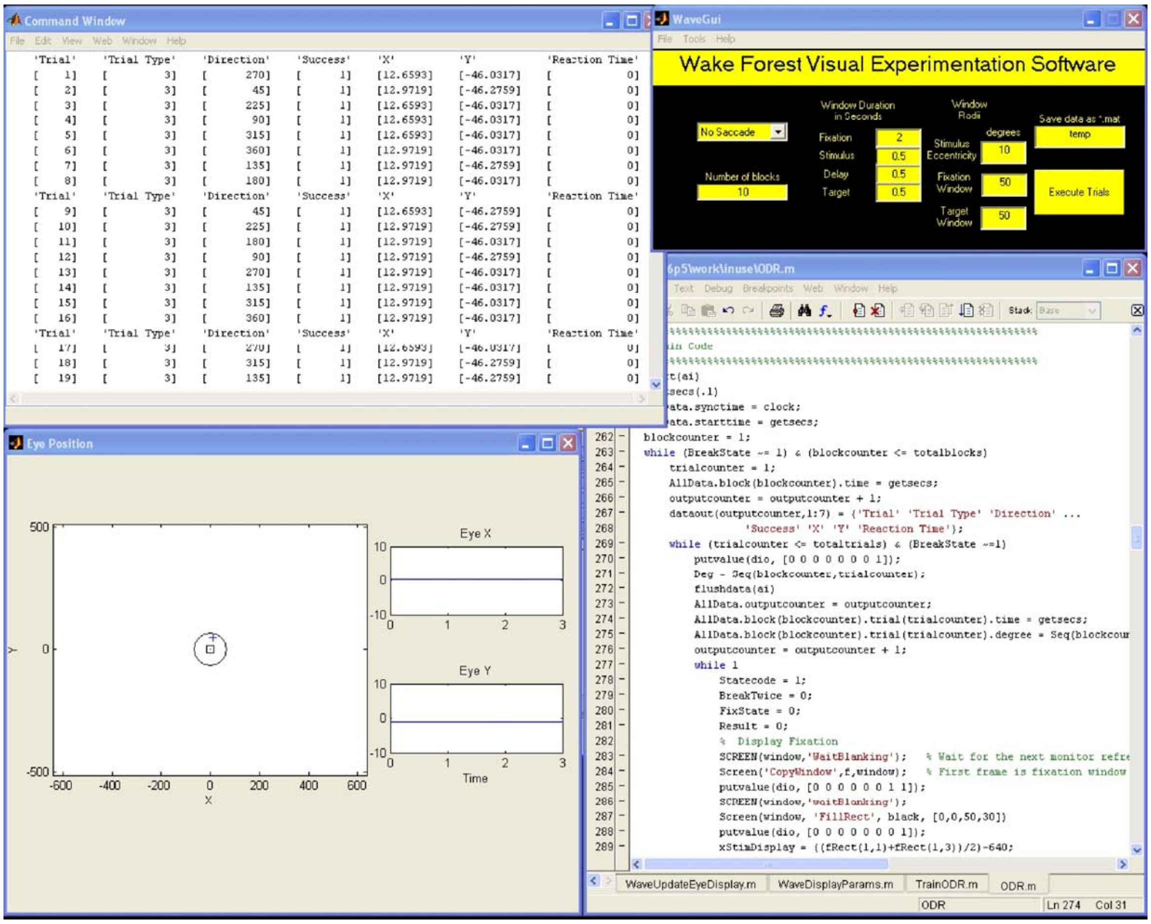
We used WaVE to implement an oculomotor delayed-response task (ODR), replicating the behavioral paradigm used in previous neurophysiological experiments (Constantinidis et al., 2001). Each behavioral trial begins with the appearance of a fixation point that the subjects need to foveate. Eye position must be maintained within an (invisible) 2° window around the fixation point, for the remainder of the trial. While the subject is looking at the fixation point, a cue stimulus is briefly flashed for 0.5 s, followed by a delay period of 3 s. The cue may appear at one of eight locations around the fixation point, selected randomly in each trial. After the end of the delay period, the fixation target turns off, instructing the subject to move his eyes to the location of the remembered cue stimulus on the screen. WaVE monitors eye position, randomly selects a cue location, displays the fixation point and cue stimulus, checks whether the subject shifted his eyes correctly and provides audio feedback and reward depending on the successful completion of the trial.

### 2.2.2. *Visual display*

Visual stimuli are constructed by drawing onto off-screen windows which are then copied onto the main window when they need to be displayed. The Psychophysics Toolbox functions allow off-screen windows to be created in memory, for instance containing the fixation point alone, or the fixation point and cue stimulus. More complex stimuli, consisting of bitmap images or movies, may also be constructed in this way (Gold and Shadlen, 2003). Precise timing of the display is achieved with the “waitblanking” Psychophysics Toolbox function, which synchronizes the screen display with the refresh cycle of the monitor.

### 2.2.3. *User interface*

The user interface allows the experimenter to easily control all task parameters and monitor the subject’s performance. During the execution of each trial, the experimenter can view the actual contents of the subject’s display (by splitting the signal of the primary monitor). An additional monitor (Experimenter Display in Fig. 1) is used to control the task and monitor the subject’s progress and performance. WaVE uses a graphical user interface that makes it easy to input and change behavioral parameters of the task, such as the eccentricity of the cue stimulus, and the duration of the cue presentation and the delay period. Additional parameters can easily be incorporated in the menu. The full set of parameters that define the behavioral trial can be saved and later retrieved from a file. The graphical user interface was constructed with the Matlab “GUIDE” (GUI Design Envrionment) function



```
    'Trial'    'Trial Type'    'Direction'    'Success'    'X'          'Y'           'Reaction Time'
    [    1]    [          3]   [       270]   [      1]   [12.6593]    [-46.0317]    [             0]
    [    2]    [          3]   [        45]   [      1]   [12.9719]    [-46.2759]    [             0]
    [    3]    [          3]   [       225]   [      1]   [12.6593]    [-46.0317]    [             0]
    [    4]    [          3]   [        90]   [      1]   [12.6593]    [-46.0317]    [             0]
    [    5]    [          3]   [       315]   [      1]   [12.6593]    [-46.0317]    [             0]
    [    6]    [          3]   [       360]   [      1]   [12.9719]    [-46.0317]    [             0]
    [    7]    [          3]   [       135]   [      1]   [12.9719]    [-46.2759]    [             0]
    [    8]    [          3]   [       180]   [      1]   [12.9719]    [-46.0317]    [             0]
    'Trial'    'Trial Type'    'Direction'    'Success'    'X'          'Y'           'Reaction Time'
    [    9]    [          3]   [        45]   [      1]   [12.6593]    [-46.2759]    [             0]
    [   10]    [          3]   [       225]   [      1]   [12.9719]    [-46.2759]    [             0]
    [   11]    [          3]   [       180]   [      1]   [12.9719]    [-46.0317]    [             0]
    [   12]    [          3]   [        90]   [      1]   [12.9719]    [-46.2759]    [             0]
    [   13]    [          3]   [       270]   [      1]   [12.9719]    [-46.0317]    [             0]
    [   14]    [          3]   [       135]   [      1]   [12.9719]    [-46.0317]    [             0]
    [   15]    [          3]   [       315]   [      1]   [12.9719]    [-46.0317]    [             0]
    [   16]    [          3]   [       360]   [      1]   [12.9719]    [-46.0317]    [             0]
    'Trial'    'Trial Type'    'Direction'    'Success'    'X'          'Y'           'Reaction Time'
    [   17]    [          3]   [       270]   [      1]   [12.6593]    [-46.0317]    [             0]
    [   18]    [          3]   [       315]   [      1]   [12.9719]    [-46.2759]    [             0]
    [   19]    [          3]   [       135]   [      1]   [12.9719]    [-46.2759]    [             0]
```

No Saccade

Number of blocks: 10

Window Duration in Seconds: Fixation 2; Stimulus 0.5; Delay 0.5; Target 0.5

Window Radii (degrees): Stimulus Eccentricity 10; Fixation Window 50; Target Window 50

Save data as *.mat: temp

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
in Code
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
t(ai)
secs(.1)
ata.synctime = clock;
ata.starttime = getsecs;
blockcounter = 1;
while (BreakState ~= 1) & (blockcounter <= totalblocks)
    trialcounter = 1;
    AllData.block(blockcounter).time = getsecs;
    outputcounter = outputcounter + 1;
    dataout(outputcounter,1:7) = {'Trial' 'Trial Type' 'Direction' ...
          'Success' 'X' 'Y' 'Reaction Time'};
    while (trialcounter <= totaltrials) & (BreakState ~=1)
        putvalue(dio, [0 0 0 0 0 0 0 1]);
        Deg - Seq(blockcounter,trialcounter);
        flushdata(ai)
        AllData.outputcounter = outputcounter;
        AllData.block(blockcounter).trial(trialcounter).time = getsecs;
        AllData.block(blockcounter).trial(trialcounter).degree = Seq(blockcour
        outputcounter = outputcounter + 1;
        while 1
            Statecode = 1;
            BreakTwice = 0;
            FixState = 0;
            Result = 0;
            %  Display Fixation
            SCREEN(window,'WaitBlanking');    % Wait for the next monitor refre
            Screen('CopyWindow',f,window);    % First frame is fixation window
            putvalue(dio, [0 0 0 0 0 0 1 1]);
            SCREEN(window,'waitBlanking');
            Screen(window, 'FillRect', black, [0,0,50,30])
            putvalue(dio, [0 0 0 0 0 0 0 1]);
            xStimDisplay = ((fRect(1,1)+fRect(1,3))/2)-640;
```

Fig. 2. Screen shot of the experimental display monitor indicated in Fig. 1. Windows counter-clockwise from the upper right: graphical user interface used to input parameters and launch an experiment; behavioral performance statistics window; graph window, displaying the subject's eye position (cross), relative to the fixation point (square) and the fixation window (circle); Matlab window, executing the WaVE script.

which allows the user to interactively design a graphical window and link variables and commands to it. An additional window in the experimental display (Fig. 2, top left) provides information on the subject's performance, including the number of trials executed so far and the percentage of correct responses. A Matlab graph window is used to depict the subject's eye position in real time (Fig. 2, bottom left).

Output was generated from WaVE in several ways. Sounds were delivered through a PC speaker, using the Matlab "SND" function. A device designed for the delivery of a reward was triggered at the completion of a correct behavioral trial with the MathWorks Data Acquisition "putvalue" function. Finally a TTL pulse was transmitted to the Neural Data Acquisition system, also through the "putvalue" function.

### 2.2.4. *Hardware devices controlled by WaVE*

We monitored eye position through an ISCAN, infra-red system which produced an analog signal and was connected to the Data Acquisition Board. A custom-made key box was also connected to our system in the same way. Analog signals were sampled by WaVE with the "getsample" function of the MathWorks Data Acquisition Toolbox. Analog data handled through the Data Acquisition Toolbox are buffered and can be saved asynchronously to the hard disk. We saved the data and flushed the buffer at the end of each behavioral trial with the "getdata" function. The "getsample" function served only to provide real-time, behavioral control of the task, mainly to monitor whether the subject's eye position had deviated from the fixation point, requiring termination of the trial.

### 2.2.5. *Communication with neurophysiological data acquisition*

We dedicated a second Personal Computer to the acquisition of neural data. A Matlab session running on this computer was used to display plots of neural activity, recorded in each behavioral trial. We connected the two computers through the Ethernet network and transmitted information between them using the "pnet" function of the TCP/UDP/IP toolbox. We used the function to send a text string to the TCP/IP address of our Neural Data Acquisition computer, and to a specified port. This information was read by the software of our Data Acquisition system (APM system, FHC) and saved directly to the file storing the neural data. We included fields such as the behavioral trial type, the size and location of the visual stimulus, and the subject's reaction time in a single text string which was transmitted through "pnet" and appropriately decoded by the Matlab session running on the Neural Data Acquisition computer.

## 3. Results

We evaluated the performance of the WaVE software under different configurations and running on different systems. At a minimum, we wished to ensure that WaVE could reliably display visual stimuli at a rate faster than the refresh rate of high-frequency monitors (120 Hz) and that it could sample multiple hardware devices faster than the highest resolution of eye tracking systems (1 ms). We present results only regarding the monitoring of behavioral performance; WaVE was not used to collect neurophysiological data, which were instead handled by the native software of our Neural Data Acquisition system (APM system, FHC).

The Windows XP operating system allows some control over the processing priority of the applications running on it. Using the Windows Task Manager and right clicking on the Matlab process we set its priority to "High" for the execution of our tests. A yet higher priority of "Real-Time" is available, but may make the system difficult to interrupt if it is necessary to abort execution of a Matlab script for any reason. The operating system processes may still take priority over Matlab, and it is conceivable that Windows may at times delay execution of the WaVE software, producing occasional spikes in its performance. As we document below, we never observed significant delays in the program's execution. We did however take the additional precautions of refraining from running other applications while running WaVE and of turning the computer off at the end of each daily recording session.

### 3.1. Visual display update accuracy

For the purposes of our simulations we generated visual stimuli and calculated the time lag between the Matlab function to display the screen and the time when the stimulus appeared on the monitor. This was accomplished by generating a TTL pulse immediately after issuing the command to display the screen and using a photodiode attached to the screen to register the time a stimulus appeared on the monitor. We used an 18 in., Dell Ultrascan, cathode ray tube monitor for these measurements, always displaying a stimulus on the upper left corner of the screen, which is illuminated first in each refresh cycle. Both TTL and photodiode signals were directed to our Neural Data Acquisition system, which computed the timing difference between the two. Precise timing of stimulus presentation was achieved using the "waitblanking" function, which only begins to draw on the screen when the monitor starts a new refresh cycle. The results of 1000 simulated trials are shown on Fig. 3. The time to copy the off-screen window and illuminate the upper-left corner pixels was 0.5 ms on average, and never exceeded 1 ms. The dotted line represents the refresh rate of a 120 Hz monitor (8.3 ms) for comparison. A longer delay would be expected for stimuli appearing at locations other than the upper left corner of the monitor. This lag, however, is entirely determined by the refresh rate of the monitor and the known position of the stimulus on the screen, and can therefore be readily computed and compensated for. We also tested the reliability of the "waitblanking" synchronization function itself. The required time for the function to execute should be no longer than the inverse of the monitor's refresh rate. Slower execution times would indicate that the function skips frames. In a set of 50,000 simulations and using the fastest monitor we had available, with a refresh rate of 85 Hz, we never observed a lag time longer than the redraw cycle (11.8 ms).

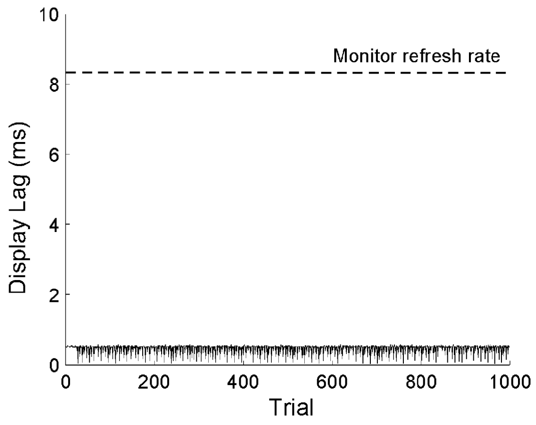


Fig. 3. Lag time between the Matlab function call displaying the screen and the actual monitor display, measured by means of a photodiode. Each data point represents the lag time for stimulus display in one behavioral trial. Dotted line represents the refresh interval of a 120 Hz monitor, for comparison.

### 3.2. Behavioral-parameter sampling resolution

We proceeded to test a number of devices that might be sampled on-line during the execution of a behavioral task. These included a standard mouse and keyboard connected to the Behavioral Control Computer. Analog inputs were directed to the Data Acquisition Board, including eye position signals scanned through an infra-red eye monitor, and a custom-built key that generated an electrical pulse. Using the MathWorks Data Acquisition Toolbox, we set the frequency rate at which the analog board sampled its inputs to 1 ms. We also estimated the computing time required to plot the real-time eye position onto the experimenter display screen. In each case, WaVE sampled continuously a device for a predetermined interval (e.g. sampled the eye position over the 0.5 s of the stimulus presentation on the screen) and computed the sampling rate achieved.

Average results from 1000 simulations are shown in Fig. 4A. Each data point in the figure represents the sampling resolution achieved during a 0.5 s long interval. The sampling resolution was found to be less than 0.3 ms, in each

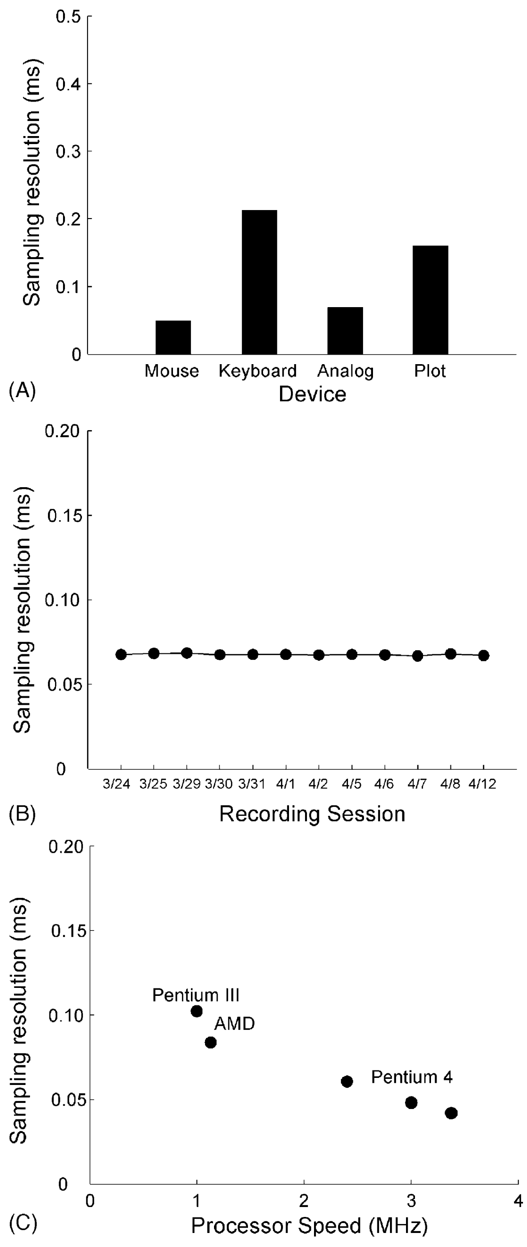


Fig. 4. (A) Average sampling resolution achieved for the control of experimental devices. Results of 1000 simulations, each 0.5 s long. Points represent sampling resolution for the mouse and keyboard, connected directly to the behavioral control computer, for analog eye position data, sampled through the data acquisition board, and the time required to refresh a plot indicating the subject's eye position on the experimenter's display screen. (B) Sampling resolution achieved for the analog data acquisition alone, in different recording days. (C) Sampling rate achieved by WaVE in a variety of computers. Only the mouse was sampled for each of these simulations. The computer's processor speed is represented in the abscissa. The type of processor is indicated next to the data point. All systems ran the Windows operating system (XP or 2000).

case. We also considered the possible interaction of different devices and calculated the time required for multiple devices to be sampled, one after the other. Fig. 5 revealed that even when all four devices were accessed in sequence, the shared sampling resolution did not exceed 0.4 ms. Sampling the keyboard is not necessary for neurophysiological applications, therefore the effective sampling resolution making use of only the analog data acquisition, mouse and eye-position plot was estimated to be 0.16 ms in our system. This is approximately six times faster than the fastest available eye-position sampling rate, achieved by means of a scleral search coil (Judge et al., 1980). We should note that this sampling rate only refers to the intrinsic speed of execution of our Matlab script—the hardware devices being sampled may have slower refresh rates.

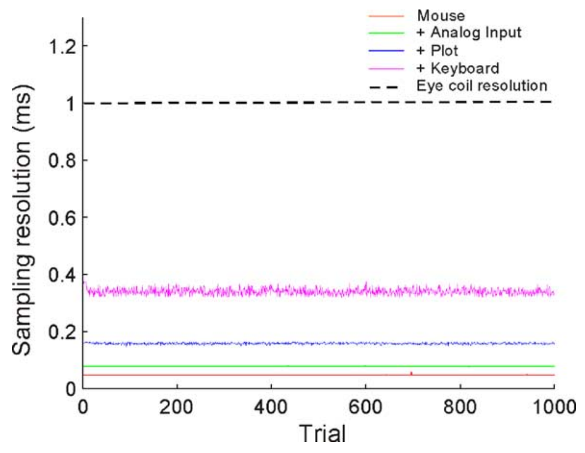


Fig. 5. Cumulative effect of sampling multiple devices. Each data point represents sampling resolution in a 0.5 s-long interval. Results for 1000 consecutive trials are shown. Successive lines represent the sampling resolution achieved by adding one device to all the previous ones. Dotted line illustrates the temporal resolution of a high-frequency eye monitoring system (1 ms).

We also considered that multiple analog channels may need to be sampled from the Data Acquisition Board. We found that the MathWorks Data Acquisition Toolbox functions handled data collection very efficiently, with virtually no increase in the sampling rate when we sampled a single analog channel (horizontal eye position) or when we added a second (vertical eye position) and a third analog channel (key-press signal) to the data acquisition board. The sampling resolution of the analog board was found to be unchanged at 0.069 ms for one, two and three channels being sampled, respectively. Additionally, we found the sampling resolution to be highly reliable across multiple trials (Fig. 5) and recording days (Fig. 4B).

We tested our demonstration program, which only sampled the mouse and required no specialized hardware, on several computers. We found a positive correlation between processor speed and sampling resolution (Fig. 4C), however even older computers could achieve sub-millisecond sampling rates.

In a final set of simulations we wished to test the sampling rate variance over very short time intervals. We wished to make sure that the instantaneous sampling resolution (the inverse of the interval between two successive samples) did not exceed our 1 ms benchmark (Fig. 6). Although the instantaneous sampling resolution of the analog board proved to be more variable than the average computed over the length of a

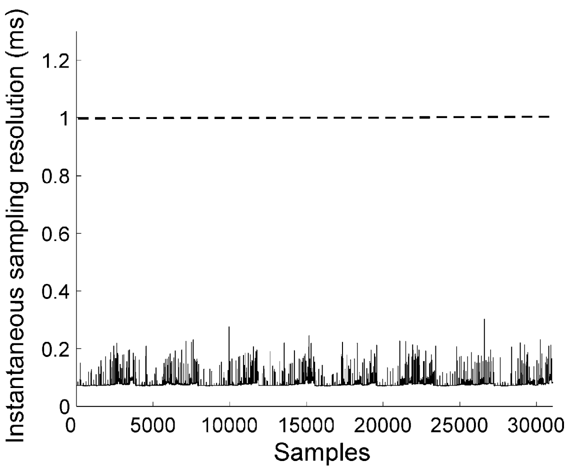


Fig. 6. Instantaneous sampling resolution for analog data acquisition. Each data point represents the time interval between two successive analog inputs being sampled continuously.

behavioral trial (Fig. 5), it never exceeded 0.3 ms. These measurements were obtained by using the system clock, which is accessible to Matlab. As an independent means of verifying the continuity of Matlab executions, we saved TTL pulses to our data acquisition system at each cycle of the loop checking the analog board for eye position. This test produced virtually identical results with those obtained using the system clock and in 375,000 iterations we never observed a sample exceeding our 1 ms benchmark. It is important to point out that even if the instantaneous sampling rate occasionally lags the output of hardware devices, this would not lead to a loss of data. Analog data sampled through the Data Acquisition Toolbox are buffered and can be retrieved in their entirety at the end of a trial. The only adverse effect of a spurious spike in the instantaneous sampling rate (which we never observed) would be a loss of real-time behavioral control over a very short period of time, e.g. a delay of 1–2 ms in detecting that eye position has deviated away from the fixation window.

## 4. Discussion

We demonstrate the implementation of computer software for the behavioral control of visual experimentation. Our system, designed entirely within the Matlab platform, provides a simple, flexible, and powerful solution for use in visual psychophysics, neurophysiology, fMRI and related applications. The Wake-Forest Visual Experimentation software makes use of mature and readily available libraries of functions to display visual stimuli, query peripherals and output signals to hardware devices and computers. Our simulation results demonstrate that WaVE achieves sub-millisecond control over these functions, without explicit optimization.

WaVE is an application developed for primate neurophysiology that can easily be customized to individual laboratory needs. Implementation of our entire behavioral task interfacing with the neural data acquisition required less than 700 lines of code. Changes in behavioral parameters, for example the duration of a stimulus, could be performed using a Graphical User Interface, without the need for changes on the Matlab script itself. Additional variables and capabilities could be added as needed. In addition to its simplicity, the system presents many advantages. It only requires a single, fairly inexpensive computer (which could be used for multiple purposes other than experimental control). We performed all our simulations on a Windows Personal Computer system, however the portability of the Matlab code allows for use on other platforms, as well. The range of device drivers available for each operating system presents the only limitation on the hardware devices that WaVE could control. The system also provided real-time graphical output to the experimenter about the progress of the behavioral trial, using the sophisticated Matlab graphic capabilities, and was capable of receiving and transmitting messages to other computers through the Ethernet network. Finally, it achieved high temporal resolution in the handling of hardware devices. We should caution that the results of our simulations are contingent on the computational requirements of the behavioral task used. Tasks requiring more intensive real-time computations during execution of a behavioral trial may slow the system's performance. Individual experimenters should evaluate the timing of their customized Matlab scripts in the manner that we have done here.

We made little effort to optimize our code for speed but opted for maximum clarity, instead. The system's performance can therefore be further increased. For example, even when multiple devices need to be accessed simultaneously, it is not necessary to refresh the subject's eye position plot on the experimenter's display screen at the maximum rate possible, or sample the mouse at sub-millisecond rates. Such functions can be performed at longer intervals, freeing up more processing time for analog data acquisition, if needed. The results of our simulations should be viewed therefore as conservative estimates of the performance that can be achieved by the system.

Although we opted to handle the Behavioral Control and the Acquisition of Neural Data by two separate computers, in principle they too could be integrated into a single computer, simultaneously running two Matlab sessions. A multiprocessor computer dedicating a processor to each function would seem well suited for such an application, which would further simplify the system's design. However, we found that monitoring both the behavioral and neurophysiological data in complex multiple-electrode recordings could be more effectively handled by two experimenters, who were better served by each controlling a separate computer, as in our setting.

## Acknowledgments

We wish to thank Bill Vaughan and Andrei Barborica for their help with hardware and software issues, and Terrence Stanford and Emilio Salinas for their comments on the manuscript. Supported by NIH training grant T32 DC00057 (T.M.) and a McDonnell-Pew Program in Cognitive Neuroscience Award (C.C.)

## References

Brainard DH. The Psychophysics Toolbox. Spat Vis 1997;10:433–6.

Constantinidis C, Franowicz MN, Goldman-Rakic PS. Coding specificity in cortical microcircuits: a multiple electrode analysis of primate prefrontal cortex. J Neurosci 2001;21:3646–55.

Gold JI, Shadlen MN. The influence of behavioral context on the representation of a perceptual decision in developing oculomotor commands. J Neurosci 2003;23:632–51.

Judge SJ, Richmond BJ, Chu FC. Implantation of magnetic search coils for measurement of eye position: an improved method. Vision Res 1980;20:535–8.

Pelli DG. The VideoToolbox software for visual psychophysics: transforming numbers into movies. Spat Vis 1997;10:437–42.