

OpenControl: A free opensource software for video tracking and automated control of behavioral mazes

Paulo Aguiar^a, Luís Mendonça^a, Vasco Galhardo^{a,b,*}

^a Instituto de Biologia Molecular e Celular (IBMC), Universidade do Porto, Portugal

^b Instituto de Histologia e Embriologia, Faculdade de Medicina, Universidade do Porto, Portugal

Received 19 April 2007; received in revised form 25 June 2007; accepted 27 June 2007

Abstract

Operant animal behavioral tests require the interaction of the subject with sensors and actuators distributed in the experimental environment of the arena. In order to provide user independent reliable results and versatile control of these devices it is vital to use an automated control system. Commercial systems for control of animal mazes are usually based in software implementations that restrict their application to the proprietary hardware of the vendor. In this paper we present OpenControl: an opensource Visual Basic software that permits a Windows-based computer to function as a system to run fully automated behavioral experiments. OpenControl integrates video-tracking of the animal, definition of zones from the video signal for real-time assignment of animal position in the maze, control of the maze actuators from either hardware sensors or from the online video tracking, and recording of experimental data. Bidirectional communication with the maze hardware is achieved through the parallel-port interface, without the need for expensive AD-DA cards, while video tracking is attained using an inexpensive Firewire digital camera. OpenControl Visual Basic code is structurally general and versatile allowing it to be easily modified or extended to fulfill specific experimental protocols and custom hardware configurations. The Visual Basic environment was chosen in order to allow experimenters to easily adapt the code and expand it at their own needs.

© 2007 Elsevier B.V. All rights reserved.

Keywords: Automated control; Behavior tests; Opensource software

1. Introduction

Many behavioral experiments in neuroscience address high-level cognitive processes and are therefore extremely complex to design and analyze. Since a vast number of variables may, and often do contribute to the outcome of the experiments, it is paramount to reduce possible external factors of variability, of which human factors are particularly noteworthy. The complexity of behavioral testing is further increased by the need to precisely quantify the core variables which have been defined as representative of the animal's behavior. These two reasons, user independent control and precise quantitative measurements, motivate the use of automated control systems for behavioral mazes. Operant protocols used in cognitive neuroscience research (Buccafusco, 2001) regularly require interaction between the animal and several mechanical or elec-

tronic actuators that include levers, nose pokes, shockers, visual or audio stimulators, and food or drink dispensers. Commercial systems from top vendors already have software and hardware packages to control all the sensors and actuators of the maze, providing the possibility to define particular experimental conditions without the need for the experimenter to have computer programming skills. However, in addition to their high cost, these control systems may also present experimental limitations: they cannot be used universally to control actuators of any manufacturer and they commonly restrict their input to analogic signals (therefore, video tracking of animal position cannot be used to control actuators). Moreover, many experimental settings require the real-time integration of the behavioral data with other methods (e.g. neurophysiological recordings, drug delivery) asking for a degree of customization that commercial control systems cannot anticipate.

Several experimental protocols require well defined constraints for the actuators activity: for example in radial arm mazes it is convenient to dispense a food pellet only when the animal advances halfway through each baited arm, which is commonly accomplished using photocell sensors; figure-eight mazes

* Corresponding author at: Instituto de Biologia Molecular e Celular (IBMC), Rua do Campo Alegre, 823, 4150-180 Porto, Portugal. Tel.: +351 22 6074900; fax: +351 22 6099157.

E-mail address: galhardo@med.up.pt (V. Galhardo).

for delayed alternation tasks or circular mazes for probing the activity of hippocampal place cells may also operate doors and feeders according to the location of the animal (Belhaoues et al., 2005; Dudchenko, 2004; Lee and Wilson, 2002); for example, it was recently described a rewarded learning task in which the animals were trained to use auditory cues of distance to goal in order to locate an unmarked area in an open-field (Bao et al., 2004). Video tracking of the animal location is therefore not only used for offline trajectory calculation but becomes a necessary element for real-time control of the maze hardware. Automated control of these types of mazes would be extremely easy to implement using a video camera instead of photocells for determining the animal position, and custom solutions of video-based control have been developed to satisfy particular needs (Pedigo et al., 2006). Both free and commercial solutions are already available for either video tracking of animal position (Noldus et al., 2001; Ramazani et al., 2007; Togasaki et al., 2005) or control of maze hardware (Zhang, 2006). However, no free simple solution exists that integrates animal tracking with maze hardware control. Even commercial solutions for this purpose are not common.

Given the number of hardware customization that some experimental protocols require, we have developed in our lab a versatile control system for behavioral mazes that allows the free integration of any electronic component deemed necessary and also includes real-time video tracking of the animal position. Three priorities guided the development of the software running this control system: it should be open-source, no expensive hardware components should be required and it should be very easy to adapt the software for specific experimental needs.

OpenControl was designed in a modular way so that every user may easily adapt the code to the needs of its particular situation. In this paper we describe the core of the software and electronic components that we have developed in order to interface with actuators from commercial manufacturers, and describe the application of the system in the control of a particular experimental protocol.

2. Materials and methods

The entire control system is made of five components: the control computer, the OpenControl software, the video camera, the maze hardware modules, and the custom electronics needed in order to use the parallel LPT1 port to control generic maze hardware (Fig. 1). Each one of these components will be described separately. Additional technical details on the setup and use of the system can be found on the webpage set for free download of the software: <http://sourceforge.net/projects/OpenControl>.

2.1. Control computer

A standard personal computer (PC) has been used to develop, test and operate OpenControl automation control system. The computer outfitted an Intel Pentium 4 processor @ 3.0 GHz and 1024 RAM. Computer configuration is not critical for proper functioning of the system, although faster processors will result

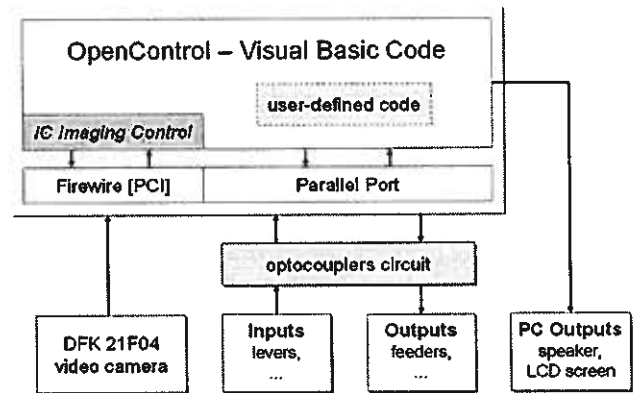


Fig. 1. Configuration setup of the OpenControl automation system.

in smoother performance of the tracking algorithm. As in all standard PCs, this computer had one parallel port available, LPT1. For Firewire support, required by our video camera (see below), we installed a generic Firewire PCI card. The computer ran the Windows XP (Microsoft) operating system and we installed Visual Basic 6.0 (Microsoft) to develop and compile OpenControl software. Notice that, once compiled, OpenControl can run in any Windows XP computer without Visual Basic 6.0 installed.

2.2. Open control software

OpenControl is written in Microsoft's Visual Basic 6 (but, if required, can easily be ported Visual Basic .Net). Visual Basic is an intuitive and popular programming language, accessible even to people who are not familiar with programming. It has the additional advantage of being contemplated on most hardware drivers through the inclusion of Visual Basic ActiveX modules or .Net components. This is the case for many digital cameras, data acquisition boards and other hardware often used in automated control. These ActiveX modules or .Net components allow a transparent and uncomplicated integration of the hardware control into the software.

The OpenControl automated control software is formed by the following components: automation control code, graphical user interface (GUI), tracking algorithm and data acquisition and communication.

2.2.1. Automation control code

OpenControl has been written in order to segregate the code for specific logic state implementations (automated control) from all other "housekeeping" code such as reading and writing data to the parallel port or running the tracking algorithm. This is accomplished by creating specific user defined routines which are aggregated in an independent Visual Basic module file named UserCode. By writing simple code into these routines (in most cases only uncomplicated IF-THEN-ELSE conditions are necessary) the user can define the web of stimuli-response, action-reaction conditions which characterize the behavioral test experiment. These routines have access to all variables relevant for control such as sensors states, actu-

ators states, animal's position, etc. For example, if one wants to start actuator 1 when sensor 2 is activated, the code is simply: "IF sensor(2).state = True THEN actuator(1).state = True".

2.2.2. Graphical user interface

The generic graphical user interface of OpenControl provides information regarding sensors states, animal position (if the tracking algorithm is running), allows the user to load and save data, among other things. In the automated system, the control actions are initiated not only by sensor events but also by user events. User events are coded on top of the graphical user interface and are usually associated with key presses or button presses. OpenControl supports key presses to override sensor or actuator activation and uses buttons to define and initiate the cascade of events associated with the beginning of an experiment or beginning of a trial. A screenshot of the default OpenControl GUI window is depicted in Fig. 2. Different experiment protocols, with different devices, may require a more specific frontend. Again, the Visual Basic environment is advantageous since it is extremely simple to design customized frontends.

2.2.3. Tracking algorithm

The OpenControl software includes a tracking algorithm capable of producing real-time estimates of the animal's position from an input video streaming. The position information is available in the code as a global variable and can be used for location monitoring or can be additionally used to set state

logic definitions in the automated control system. For example, a door or gate can be automatically opened when the rat is in its neighbourhood. The tracking algorithm uses a background image of the experimental arena without the animal, to continuously produce an image mask defining regions of occlusion. The mask is produced by comparing the present image with the background: areas where the value of the pixels, in all three colour filters (red, green and blue), exceed a predefined threshold are signalled as an occlusion area. All pixels in the occlusion area are then scored according to user-defined colour targets. This score is calculated using the Euclidean distance in RGB space between the pixel value and target colour. The scores are semi-linear in the sense that differences below a predefined threshold are zeroed. This nonlinearity is essential for the quality of the results. The position is then calculated as the centroid of the weighted region.

Two target colours can be chosen, meaning that two positions are independently calculated. This serves the purpose of setting body and head locations in an animal where a colourful spot has been painted on its head. The tracking algorithm depends on the environment light conditions and their variability. However, leaving the camera in automatic white balance settings and back-light compensation (two options commonly available in video input devices) produces consistent light and contrast conditions across time which allows the algorithm to work properly and reliably. Both thresholds for background and colour target can be changed by the user from the graphical user interface. This

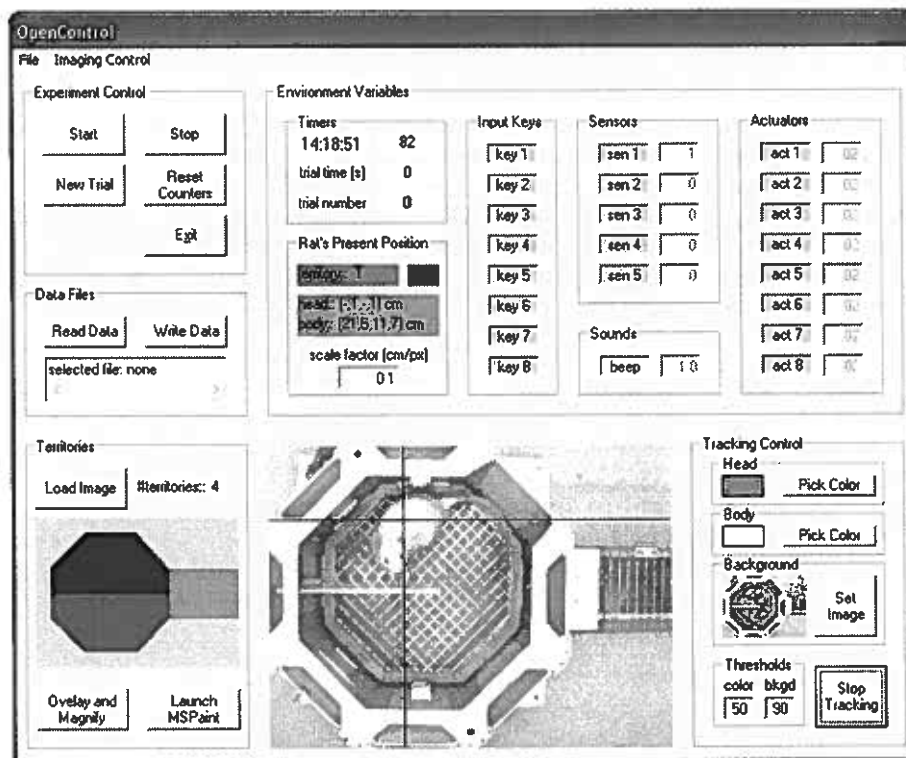


Fig. 2. Screenshot of OpenControl's basic GUI. Information regarding the state of sensors, actuators and input keys is shown in the frontend as well as other environmental variables including the animal's position. The tracking area occupies the lower half of the GUI and allows the user to set, as well as visualize, several properties of the algorithm.

allows the user to fine tune the algorithm for particular light conditions. Better tracking results are obtained when the tracked colours are very distinct, or ideally absent, from the background image colours.

Processing images require handling a large amount of data. In order to reduce the processor load, the tracking algorithm supports three simplifications with little or no effect in position detection. First, only a fraction of the video frames are used by the tracking algorithm. Unless the camera frame rate is below 10 Hz and the animal is running fast around in the arena (or the image scale is very big), this reduction does not produce noticeable changes. Secondly, only a fraction of the pixels are sampled. If the camera is in 640×480 mode, sampling 1/4 of the pixels allows the tracking to run efficiently without compromising the results. The sampling fraction limit depends on the amount of pixels involved in the animal image (where the image scale is again an important factor). Finally, the algorithm keeps track of the animal's previous position and velocity. This allows the algorithm to predict where the animal should be located in the next frame and reduce considerably the search area. Only if the animal is not found in the predicted area then the whole image will be used for tracking. Despite the simplicity of this algorithm, its performance is very good and comparable with commercial solutions. If needed, modifications and improvements to the algorithm can always be made in its open-source code.

OpenControl also allows the user to define spatial regions of interest, here named "zones" of the maze. This information is stored as an image file, with the same resolution properties as the camera image, with the different zones painted uniformly with different colours. These images can be created in any image editor and are easily loaded into OpenControl using its GUI (Fig. 2, lower left area). Using these images together with the tracking algorithm the user can define events associated with each zone without the need of complex functions of positions, or may be used for tracking analyses such as time spent on each zone of the maze, or number of entries into each one (measures commonly used in the openfield, plus or water mazes, for example).

2.2.4. Data acquisition and communication

Communication with the parallel port relies on a Windows XP dynamic link library—"inout32.dll". The code associated with sending data and reading data from the parallel port is segregated from the user code.

2.3. Video camera

The system was developed using an inexpensive DFK21F04 digital video camera (Imaging Source Europe, Bremen, Germany), fitted with a 2.8–6 mm, 1/3 in. lens (Computar, Commack, NY). The choice for this specific camera manufacturer relied on the inclusion of Visual Basic ActiveX modules (as well as .Net components) on the software drivers of their cameras (IC Imaging Control), which permits easy integration of the camera functions within the program. Another important feature behind the choice of the camera was the interchange-

ability of the lenses through a regular c-mount adaptor, so that the same camera could be adapted to record at very different distances from the animal.

2.4. Maze hardware modules

We have run the OpenControl software to control both commercial and in-house built maze actuators. Commercial modular maze components were acquired from Coulbourn Instruments (Allentown, PA, USA). Apart from the passive elements of maze construction (cages, hubs, walkways), we have tested pellet food dispensers, retractable and non-retractable levers, and nose poke sensors. In addition, we have used the OpenControl software to control custom built audio and visual stimulators, as well as a thermal stimulator built with Peltier elements. Given the usually simple connectivity schemes used in operating the commercial maze modules, we believe that no incompatibility exists between OpenControl and current maze hardware. Commercial modules for operant behavior mazes commonly belong in two classes of complexity: TTL-like controlled modules (in which the line of power supply is distinct from the line of control operational control), and switch-like modules (in which the control signal is the powering signal). Most vendors provide external power supply units for their modules, as well as voltage adaptors that convert the 5 V TTL signals from computer ports to the adequate voltage required at the control input of their modules (commonly a positive or negative 24–28 V signal). In the download page of OpenControl we also provide the schematics for an external power supply and optocoupled TTL converters that we have developed to specifically operate the Coulbourn modules.

2.5. Parallel interface hardware

With the goal of making this automated control software as general and unrestricted as possible, OpenControl was designed for the standard parallel port, LPT1. The computer parallel port interface LPT1 is a very convenient bidirectional port for communication with external devices because of: (1) its straightforward access functions under the Visual Basic programming environment and (2) the portability of the software to any computer independently of its hardware. On the other hand, the use of LPT1 port introduces two drawbacks on the OpenControl system: first, it restricts the channels of communication to a small number of input and output signals; second, it cannot be used for control with millisecond precise timing since data port access (read and write) is done at fixed sampling frequencies which may vary according with instantaneous values of CPU usage. The first drawback will affect only a few experimental situations. Even in commercial control stations, the input connections are very limited in number since most behavioral protocols commonly require only a handful of actuators. In the configuration we adopted for OpenControl, the parallel port provides 5 input bits and 8 output bits. This choice was made in order to keep the external circuit bridging the port and the devices as simple as possible (see the OpenControl webpage for schematics of this circuit). These input–output numbers can nevertheless be substantially increased in optimized configurations where the inputs

can go up to 13 bits and the outputs up to 12 bits. These configurations require, however, complex external circuits which we choose do not use in this paper (but see OpenControl website for suggestions of implementation).

The second drawback, regarding time precision issues, is not imputable to the parallel port alone but to its control through Visual Basic. Best performance in an automated control system is achieved using background or asynchronous operation. However, the parallel port under Visual Basic (as well as most low and medium range data acquisitions boards) does not support this mode. Instead, all input bits have to be read synchronously in what is called polling mode. In polling mode OpenControl defines a sampling frequency for checking the status of the input bits. An external S-R flip-flop circuit (latch circuit) ensures that every sensor activation (as narrow as 10^{-7} s) is preserved until acquired by the following LPT1 read command. While the time resolutions appropriate for most behavioral tests are guaranteed, millisecond precision is not possible. See Section 4 for a complete analysis of the time precision and time resolution of OpenControl.

2.6. Other hardware configurations

Where required, other hardware configurations can be used without introducing significant changes into OpenControl. A component with noteworthy significance is the data acquisition interface. If millisecond precision is required, the parallel port interface can be replaced with a specialized data acquisition board. Many boards come with ActiveX or .Net components making their integration into the Visual Basic environment straightforward.

3. Results

The OpenControl system has been used in our laboratory in the control of several protocols, including the *Rodent Gambling Task* reported elsewhere (Pais-Vieira et al., 2007). It has also been used in experiments involving tethered rats with chronically implanted multielectrodes for simultaneous recording of behavioral and neural activities (Pais-Vieira et al., unpublished results). The use of optocoupling circuitry in the bidirectional connections through the parallel port prevented the occurrence of electrical noise generated by the control computer.

In the above-mentioned Rodent Gambling Task (Pais-Vieira et al., 2007), the generic code of the OpenControl software was customized to operate a novel behavioral protocol designed to probe the role of the orbitofrontal cortex on an emotional decision-making task. Further details about the results and relevance of the study can be found in the original paper. Here we will describe the use of the OpenControl system to its operation as a simple example of the customization easily attainable by this system.

The hardware of the behavioral apparatus consisted in a rectangular chamber connected to an octagonal hub through a motorized guillotine door. The hub was partially divided into two adjacent chambers by a 15 cm protruding opaque Plexiglas panel. Each half of the hub contained one non-retractable

lever and one automated pellet dispenser. All the above modules were acquired from Coulbourn Instruments. In addition we also installed in the hub a ceiling light and an audio speaker. Hence, the OpenControl received two inputs, one from each lever, and generated five outputs: two pellet feeders, guillotine door, ceiling light and the audio speaker (driven directly from the computer audio card). Coulbourn modules are powered by -28 V and therefore we built a power supply unit for that voltage that is able to provide 3 A (circuit schematics provided in the OpenControl download page).

Briefly, the experimental protocol of the Rodent Gambling Task consisted in 90 trials of 20 s that followed the same sequence: first, the animal was placed in the entrance chamber; after a random period of 5–10 s, the ceiling light switched on, a brief audio tone signaled the begin a new trial and the guillotine door opened allowing access to the hub; the first lever the animal pressed generated the correspondent reward according to the contingency table (see below), and any further press had no longer effect; when the trial time limit was reached the ceiling light was switched off, and the animal was placed back in the entrance chamber.

The OpenControl interface was adapted so that relevant information about the progress of the task would be immediately presented to the experimenter. Since tracking the location of the animal was not relevant to this particular task, the video was only used for remote observation of the animal. The main OpenControl Visual Basic form specifying the GUI was modified so that only the status of the sensors, actuators and counters used in the task would appear in the screen (Fig. 3). Since each experimental session comprised 90 trials of fixed duration, several buttons were added to the GUI: the *BeginSession* button initiates a session; the *BeginTrial* button initiates each trial; the *EndSession* button terminates the session and records the experimental data to a predefined file. Additionally, a series of buttons to remotely operate the feeders, guillotine door, audio tone and ceiling light were introduced in order to help the experimenter during the autoshaping training of the animals, but were not used during the experimental session. The remaining items in the GUI refer to the timers and counters used in the task and that are displayed only for feedback of the experimenter.

The core of the task relies on the fact that choices between the left or right levers led to different rewards in size and probability: one lever attributed a one pellet reward with 0.8 probability (*low risk lever*), while the other attributed a three pellets reward with only 0.3 probability (*high risk lever*). Rewards per lever followed a pseudo-random pattern that was kept fixed for every animal so that the sequence of rewards in each lever would always be the same. This contingency table was defined in a separate configuration file read by OpenControl at the beginning of a session. The few specific rules necessary to control the modules were inserted in the *UserCode* module file, and they are detailed in Table 1 (additional Visual Basic variables were used to keep track of the trial-by-trial events, but those were omitted here for simplicity). At the end of the experimental session, trial-by-trial events and the final value of the relevant counters were saved in tabbed text format importable in MATLAB for further analyses.

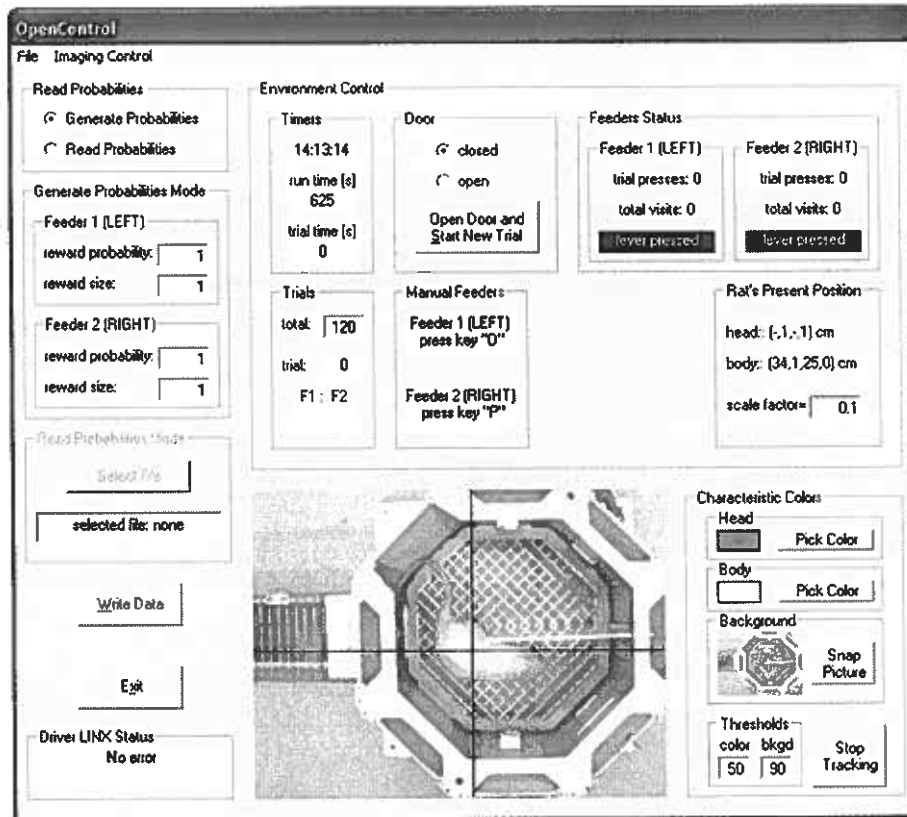


Fig. 3. OpenControl's implementation for a specific behavioral test. Some buttons were added to the OpenControl's basic GUI and all unnecessary controls were removed. The process of redesigning the GUI is completely straightforward in Visual Basic.

3.1. Performance analysis of open control

To quantitatively assess the performance of the software several tests were carried out. These tests included assessment of sampling resolution, time delays and time jitter. All tests were performed with a TDS 1002 two-channel digital storage oscilloscope with 60 MHz sampling frequency (Tektronix, Courtaboeuf, France).

The sensors sampling resolution can be defined in OpenControl up to 1 KHz but there is no *a priori* guarantee of the precision or accuracy attained: since the sampling is done in foreground (synchronously), very high resolutions will not be respected, especially if the CPU is under high loads.

To define an upper bound for the sampling resolution, given the computer configuration mentioned earlier, we have set an artificial setup where OpenControl sensors were driven by an

Table 1
Logical operations coded in the UserCode module

Event	Actions
BeginSession button	Reset all timers and counters; start timer <i>SessionTime</i> ; set button <i>BeginSession</i> as OFF; set buttons <i>EndSession</i> and <i>BeginTrial</i> as ON
BeginTrial button	Increment counter <i>TrialNumber</i> ; set buttons <i>EndSession</i> and <i>BeginTrial</i> as OFF; set timer <i>TrialWait</i> to random value between 5 and 10 s; start countdown
Timer <i>TrialWait</i> reaches "0"	Ceiling light ON; sound tone; open guillotine door; set timer <i>TrialTime</i> to "20"; start countdown; set counter <i>LeverAvailable</i> to "1"
Sensor <i>LeverRight</i> ON and <i>LeverAvailable</i> equals "1"	Set counter <i>LeverAvailable</i> to "0"; increment counter <i>LeverRight</i> ; if reward is due in this trial, set <i>RightFeeder</i> as ON; if reward is due in this trial, increment counters <i>TotalPellets</i> and <i>PelletsRight</i>
Sensor <i>LeverLeft</i> ON and <i>LeverAvailable</i> equals "1"	Set counter <i>LeverAvailable</i> to "0"; increment counter <i>LeverLeft</i> ; if reward is due in this trial, set <i>LeftFeeder</i> as ON; if reward is due in this trial, increment counters <i>TotalPellets</i> and <i>PelletsLeft</i>
Timer <i>TrialTime</i> reaches "0"	Set counter <i>LeverAvailable</i> to "0"; ceiling light off; close guillotine door; set buttons <i>EndSession</i> and <i>BeginTrial</i> as ON
EndSession button	Proceed to <i>WriteData</i> section; reset all timers and counters; set button <i>BeginSession</i> as ON; set buttons <i>EndSession</i> and <i>BeginTrial</i> as OFF

external square wave generator (3001 Signal Generator-BK Precision, CA, USA). The maximum sampling rate was defined as the limit frequency where, reliably, no pulse from the external signal generator was lost. This test was performed with and without the tracking algorithm and the results have shown that OpenControl is capable of maintaining a reliable polling rate well above 50 Hz. However, higher frequencies affect the precision of the pulses. At a polling rate of 50 Hz, and without performing tracking, OpenControl is capable of maintaining very stable periods with sub-millisecond jitter. With the tracking algorithm running, the maximum recorded jitter was 2 ms (0.820 ± 0.733 ms; 100 recordings; error is the standard error). We thus recommend using 50 Hz as the maximum polling frequency for OpenControl.

In addition to the sampling resolution, we assessed the time precision of the system. To do that we defined two automation tests with very simple logical conditions for an actuator: in test A, the actuator should be immediately activated when a sensor was pressed; in test B, the actuator should be immediately activated when the tracked position entered a specific region. In both tests, the maximum delay measured was 8 ms, with most of the pulses arriving just in time (JIT).

All these values reported in these section all well above the precision requirements for behavioral tests, making OpenControl a robust solution as an automation system.

4. Discussion

We presented a software solution for automated control of behavioral tests. The OpenControl software, including the tracking algorithm, is freely available under the GNU general public license (<http://www.gnu.org/licenses/gpl.html>). We expect that our program may be valuable for laboratories wanting to have full access and control of the automation software. Despite being written under general assumptions, our program can be easily modified and expanded to face other needs. The process of developing a control program is greatly enhanced by having the support of an existing solution. We provide a program which can be legally reverse engineered.

It was our goal to have OpenControl running in a standard personal computer with an inexpensive digital camera and using the standard parallel port for data communication. This option avoids the costs of a dedicated data acquisition board and, as we have shown, did not impose strong limitations on experimental accuracy. Behavioral tests typically require a time precision on the order of hundreds of milliseconds which is well within the capabilities of this system (under Windows XP and Visual Basic). The number of maze elements and the time precision attained by the parallel port is perfectly compatible with the majority of the behavioral experimental setups. We have also demonstrated the additional advantages of using a standard PC to control behavioral experiments instead of a proprietary constrained hardware, since the logical control of the maze hardware

may use all the computer functions as interactive tools under OpenControl: real-time video tracking may be used to act on the maze, the experimenter is able to remotely operate each actuator via the keyboard, and in addition to the digital output provided by the parallel port both the computer display and the sound card can be used as analogic outputs to present visual stimuli, drive external hardware, or even function as touch screen sensors (Bussey et al., 2001; Izquierdo et al., 2006). Finally, we hope that the ease of configuration permitted by OpenControl boosts the development of customized protocols for behavioral research.

Acknowledgements

Supported by Fundação para a Ciência e Tecnologia (FCT), Grants POCI/SAU-NEU/63034/2004, BPD/26198/2006, and Bial Foundation, Grant 84/04. We also acknowledge Miguel Pais-Vieira for user-feedback during the development of the software.

References

- Bao S, Chang EF, Woods J, Merzenich MM. Temporal plasticity in the primary auditory cortex induced by operant perceptual learning. *Nat Neurosci* 2004;7:974–81.
- Belhaoues R, Soumireu-Mourat B, Caverni JP, Roman FS. A novel experimental paradigm for studying cognitive functions related to delayed response tasks in mice. *Brain Res Cogn Brain Res* 2005;23:199–206.
- Buccafusco JJ. *Methods of behavior analysis in neuroscience*. Boca Raton: CRC Press; 2001.
- Bussey TJ, Saksida LM, Rothblat LA. Discrimination of computer-graphic stimuli by mice: a method for the behavioral characterization of transgenic and gene-knockout models. *Behav Neurosci* 2001;115:957–60.
- Dudchenko PA. An overview of the tasks used to test working memory in rodents. *Neurosci Biobehav Rev* 2004;28:699–709.
- Izquierdo A, Wiedholz LM, Millstein RA, Yang RJ, Bussey TJ, Saksida LM, et al. Genetic and dopaminergic modulation of reversal learning in a touchscreen-based operant procedure for mice. *Behav Brain Res* 2006;171:181–8.
- Lee AK, Wilson MA. Memory of sequential experience in the hippocampus during slow wave sleep. *Neuron* 2002;36:1183–94.
- Noldus LP, Spink AJ, Tegelenbosch RA. EthoVision: a versatile video tracking system for automation of behavioral experiments. *Behav Res Meth Instrum Comput* 2001;33:398–414.
- Pais-Vieira M, Lima D, Galhardo V. Orbitofrontal cortex lesions disrupt risk assessment in a novel serial decision-making task for rats. *Neuroscience* 2007;145:225–31.
- Pedigo SF, Song EY, Jung MW, Kim JJ. A computer vision-based automated Figure-8 maze for working memory test in rodents. *J Neurosci Meth* 2006;156:10–6.
- Ramazani RB, Krishnan HR, Bergeson SE, Atkinson NS. Computer automated movement detection for the analysis of behavior. *J Neurosci Meth* 2007;162:171–9.
- Togasaki DM, Hsu A, Samant M, Farzan B, DeLanney LE, Langston JW, et al. The Webcam system: a simple, automated, computer-based video system for quantitative measurement of movement in nonhuman primates. *J Neurosci Meth* 2005;145:159–66.
- Zhang F. SuperState: a computer program for the control of operant behavioral experimentation. *J Neurosci Meth* 2006;155:194–201.