Behav Res (2017) 49:13–23
DOI 10.3758/s13428-015-0694-9

CATOS (Computer Aided Training/Observing System): Automating animal observation and training

Jinook Oh[1] · **W. Tecumseh Fitch**[1]

Published online: 7 January 2016

Abstract In animal behavioral biology, an automated observing/training system may be useful for several reasons: (a) continuous observation of animals for documentation of specific, irregular events, (b) long-term intensive training of animals in preparation for behavioral experiments, (c) elimination of potential cues and biases induced by humans during training and testing. Here, we describe an open-source-based system named CATOS (Computer Aided Training/Observing System) developed for such situations. There are several notable features in this system. CATOS is flexible and low cost because it is based on free open-source software libraries, common hardware parts, and open-system electronics based on Arduino. Automated video condensation is applied, leading to significantly reduced video data storage compared to the total active hours of the system. A data-viewing utility program helps a user browse recorded data quickly and more efficiently. With these features, CATOS has the potential to be applied to many different animal species in various environments such as laboratories, zoos, or even private homes. Also, an animal's free access to the device without constraint, and a gamified learning process, enhance the animal's welfare and enriches their environment. As a proof of concept, the system was built and tested with two different species. Initially, the system was tested for approximately 10 months with a domesticated cat. The cat was successfully and fully automatically trained to discriminate three different spoken words. Then, in order to test the system's adaptability to other species and hardware components, we used it to train a laboratory rat for 3 weeks.

Keywords Automated animal training · Animal welfare · Gamification · Automatic system · Open-source system

✉ Jinook Oh
jinook.oh@univie.ac.at

W. Tecumseh Fitch
tecumseh.fitch@univie.ac.at

[1] Department of Cognitive Biology, University of Vienna, Althanstrasse 14, 1090, Vienna, Austria

Introduction: automated animal training and testing

Studies of animal behavioral biology often involve considerable human resources, time, and data storage, for example of video recordings, during animal observation and training. For example, continuous video recording may be required when monitoring nocturnal behaviors of a certain species and documenting specific events, which may occur rarely and irregularly. Also, certain experiments require a prolonged training period, which can take more than a year. Experiments often require animals to respond reliably to certain stimuli, which may not be an easy task. Therefore, training may require a long time before the subject is ready to be tested. Additionally, long periods of training by humans can introduce unintended cues and biases for the animals undergoing training.

An autonomous system for observing and training animals could thus save human resources and reduce the amount of data that needs to be recorded, processed, and stored. The reduced amount of data can also minimize the amount of human resources needed, such as for the inspection and maintenance of large volumes of data. There have been several attempts to build automated observation or surveillance systems for biological purposes (Kritzler et al., 2008), or security (Bellotto et al., 2009; Vallejo et al., 2009). There are also many commercial products available for surveillance systems, with varying degrees of automation or artificial intelligence. However, the automation and intelligence of each system is case specific, and most of them are not applicable to animal observation/training situations. Even when these systems can be applied, considerable adjustments are usually required, entailing technical difficulties and higher costs.

In the case of prolonged training, an automated system can save human resources and eliminate potential cues and biases introduced by humans. Training with an automated system is an extension of traditional operant conditioning chambers, and many modern and elaborated versions have been developed and used in this context (Markham et al., 1996; Takemoto et al., 2011; Kangas & Bergman, 2012; Steurer et al., 2012; Fagot & Bonté, 2010; Huber et al., 2015).

CATOS (Computer Aided Training/Observing System), the system we have developed in this study, differs from most of previous systems in several respects. First, the software of CATOS is open source, including its dependencies on other libraries. Second, CATOS uses computer vision for automatic video condensation. Third, an animal voluntarily participates in training and test sessions to obtain a reward. Thus, animals do not have to be forced to participate at specific times nor moved to a separated space. All of these functions are well integrated into a single package, storing and time-stamping data in a format designed for scientific purposes.

During a session, CATOS runs for many hours without requiring the presence of human trainer or experimenter. It runs for eight or more hours because the system waits for animals to interact with it voluntarily. Despite the long operating hours, its continuous video recordings are automatically condensed by the selective recording of interactions between the animals and the device. The training regime with CATOS can be composed of multiple phases with increasing difficulty levels starting from a familiarization phase. In each phase, an animal's specific interaction with the device, which is pre-defined by the user, leads to a food reward, encouraging the animal to interact with the device. A trial consists of detecting an animal via the computer vision package, presenting a stimulus, acquiring an animal's response behavior with an input device, and actuating a positive feedback including a food reward when the animal's behavior was correct.

We implemented several advantageous features especially for prolonged animal training. CATOS tries not to record any unnecessary data by detecting the animal's presence in a region of interest and processing only relevant information with a set of condition-action rules (Russell & Norvig, 2009). It also has separate data-viewer software, which shows one JPEG image summary for each recorded video (Fig. 6). The automatic data condensation and the data viewer's image summary help the user save time and effort when reviewing the collected data. The implementation of different phases with increasing levels of difficulty, in which the animal's interaction which can potentially lead to a food reward without any considerable negative feedback, was aimed to be part of enrichment plan for the animal as an animal training "gamification" (Deterding et al., 2011).

The above features of CATOS and its inexpensive construction cost due to the free open-source software and Arduino-based open-system hardware, which CATOS relies on, makes the system suitable for various purposes and locations such as a laboratory, a zoo, and even a private home.

Related work

While there are no systems precisely using the open-source/open-hardware approach of CATOS, numerous system exist using commercial hardware and/or software to analyze animal behavior. Kritzler et al. (2008) published an indoor mouse tracking system, mainly using RFID technology, but also partially using computer vision. Although this is not a training/testing system, it is a good example of research tracking animal subjects for long period, offering useful scientific data.

Kangas and Bergman (2012) used a touchscreen apparatus to test squirrel monkeys. This system requires capturing and/or isolating of an individual animal in a separate chamber for training and testing, which can lead to several problems: (a) it may be unethical for some species, depending on the capturing process, (b) the subject's attention could be lowered, or arousal increased, especially when the species is social and the subject lost visual/auditory contact with its group, (c) human monitoring and possible interference throughout the procedure is required when an animal is captive, therefore, it is time consuming for researchers.

Closer to the goals of CATOS is the system developed by Fagot and Paleressompoulle (2009), the Automatic Learning Device for Monkeys (ALDM). ALDM does not require capturing/isolating animals, and relies on recognition of an individual animal's RFID tag. A similar system is that of Huber et al. (2015), the Automatic Learning Device for Birds (ALDB). This device also starts a session when an individual's RFID tag is recognized. These two works and Kritzler et al. (2008) all used RFID technology. While it is affordable and relatively easy to implement, there are a few noteworthy constraints of RFID technology when applied to multiple animal species. First, its reading distance varies to large extent depending on the type of RFID tags and reader.

For the purpose of tracking animals, passive tags are favored (light weight and small size) to avoid burdening an animal with an active type tag with batteries. For reliable detection of a passive tag, the tag must be placed in quite close proximity to the reader (Roberts, 2006), about a few centimeters in practice. Second, readers cannot read two tags at the same time. Both together lead to a procedure, in which an individual animal subject must be trained to put their body part with the tag close to an antennae of the reader and/or goes though certain narrow passage, which is equipped with an antennae. This type of procedure may not be feasible for many animal species.

Another important issue with these previous systems is that most of them used commercial software to run the training/testing session, and none of them published their software as open source. Furthermore, how the hardware and electronic components were implemented and integrated with their software was not mentioned in most studies. Therefore, there is little chance of involvement of an active user community to freely use, modify, and update the software and hardware.

CATOS is designed to fill this gap, building on these previous achievements, but giving full specifications of hardware and system integration along with open-source hardware. Currently, the number of people with programming skill is increasing rapidly, and many researchers are already exposed to a certain degree of programming and scripting with various programming languages and platforms such as Python, Matlab, R, and so on. Also, inexpensive, easily programmable electronic products are available as well, like the Arduino chip used in this study. Our goal with the CATOS project is to initiate the public sharing of open-source software and hardware integration, so that many other researchers can reproduce previous methodologies, improve them, or develop entirely new and innovative tasks in an open and inexpensive manner.

Testing CATOS with animal subjects

For better understanding of the technical description of the system, we first describe two test cases with animal species, followed by the detailed description of CATOS itself.

Testing CATOS with two animal species was conducted using two different prototype devices (see Fig. 1). The first prototype had three pushbuttons as a main input device. This version was tested with a pet cat for about 10 months. The main input device for the second prototype was a touchscreen. This version was tested with a laboratory rat for about 3 weeks. These two prototypes were approximately the same in functional architecture but differed in sensor devices and appearance due to structural adaptations in order to protect the hardware.

Testing with a cat

We initially tested CATOS with a 2.5-year-old female pet cat originally obtained from an animal shelter at the age of 2 years. The cat had not been exposed to any form of training since its adoption until the beginning of the training for this study in October 2012. Training was conducted every day for over 10 months in the cat's home. The main experimental room was 5 m long and 2 m wide.

Each daily session lasted between 8 and 12 h, depending on the animal's current training state. A sound from a pool of possible trial sounds was randomly played with a random inter-trial interval. The possible trial sounds were different in each training phase, and are described further below. If there was no reaction after playback, then it was not

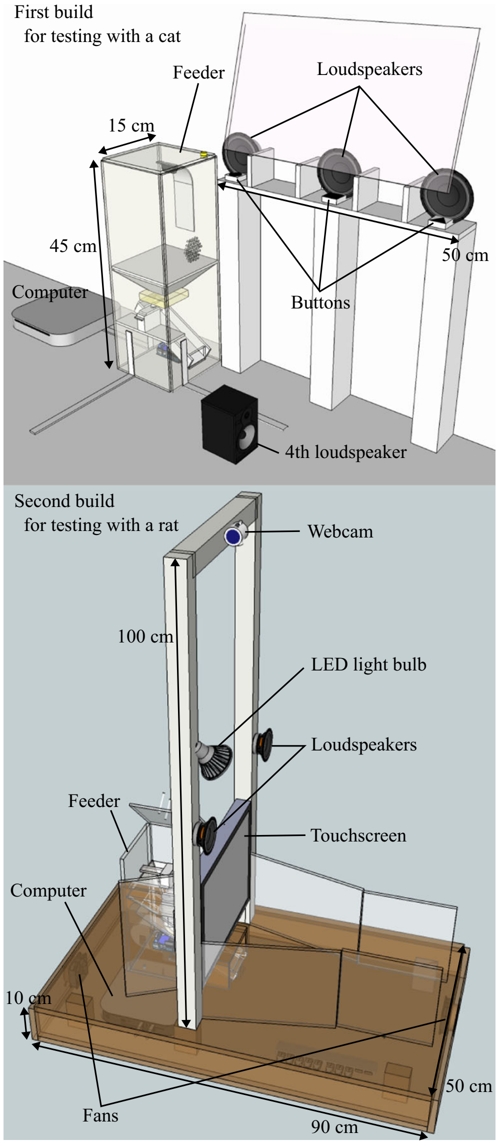


Fig. 1 Two prototype devices used to test CATOS with animal subjects

recorded as a trial in the result CSV file. Possible reactions were motion detection or button presses within the duration of the trial, which could be between 20 and 60 s, depending on the training state. If there was a reaction within the duration of the trial, then the trial officially started, which entailed logging and writing a trial entry in the result CSV file. At the end of each daily session, additional food was given to the animal, depending on the amount of food dispensed, which was indicated in the result CSV file. Also, the animal was weighed periodically through the testing period, to ensure it maintained a stable weight.

The cat first had several preliminary training phases to be familiarized with a feeder, and obtaining food from a feeder by pressing a button. Then training started, intended to associate a specific sound with pressing a specific button when multiple buttons were available. The trial duration was 60 s. Three buttons were implemented, and three sound stimuli were used, each corresponding to a specific button. The sound stimuli were recordings of the experimenter saying three Korean words, ‘Dong-g-ran’, ‘Sam-gak-hyeong’, and ‘Ne-mo-nan’, meaning round, triangular, and rectangular, respectively. The recordings had a similar duration and mean intensity, with durations of 1.35, 1.3, and 1.4 s, respectively, and mean intensities were all 64.5 dB. However, these sounds were quite different in terms of the prosody. This task was difficult for the cat to learn due to the fact that she could obtain enough food by randomly pressing one of the three buttons if she participated in a sufficient number of trials in a day. In order to facilitate the association between sounds and buttons, three loudspeakers were installed and were assigned to specific sound stimuli. In this method, each loudspeaker was located directly behind each button, playing only the sound stimulus corresponding to that button. After more than 3 months of training with three loudspeakers, a fourth loudspeaker was introduced that played any of the three sound stimuli. Figure 1 shows the placements of the loudspeakers. Over the course of a month, we gradually increased the frequency with which sounds were played over the fourth speaker relative to the other three speakers, in an attempt to train the cat to use the acoustic differences between the sounds, rather than the location of the sounds, to choose the button.

Although preliminary training phases were easily completed in several weeks, the training of the discrimination task lasted about 7 months. Within a month, the cat used all three buttons to obtain food, but did not learn to associate three sounds with three buttons. After three loudspeakers were added behind buttons to provide a spatial cue as described in the previous ‘Training procedure’ section, the correct response rate rose to more than 80 %, but when the positions of loudspeakers were changed to about 60 cm above the buttons, the correct response rate dropped to less than 60 %. With these results, we suspected the cat used only the location of the sound source as a cue, rather than the different prosody of each sound. After a fourth loudspeaker was introduced as described in the previous ‘Training procedure’ section, the three speakers were deactivated, and all trials played all three sounds from the fourth speaker. The cat’s performance on the sound discrimination task in these trials including the fourth speaker is depicted in Fig. 2, which shows a gradual increase in performance over about 2 months. The cat showed a correct response rate of more than 70 % in the final month. This protracted 10-month training period exemplifies a core virtue of the CATOS system—it required minimal time from the human experimenter, while providing the cat adequate time to learn the task based only on positive reward and voluntary participation in its own home.

Testing with a rat

A slightly modified version of the system was subsequently tested in a laboratory environment with an approximately 2-year-old male rat for several weeks. Here, our goal was simply to demonstrate that CATOS could be successfully adapted to another species and environment. The structure of sessions and trials was as same as in the first CATOS version tested with the cat: the goal was to train the rat to interact with the system. The training procedure was also similar to the preliminary training phases of the first test with the cat. The main difference was that a touchscreen was used for the rat instead of the physical push button array we implemented for the cat. While the cat could always press a button, the touchscreen only displayed a target image immediately after a sound stimulus was played for 30-s duration.

In this test, our main goal was to assess the system’s suitability to another species. This was successful: the rat

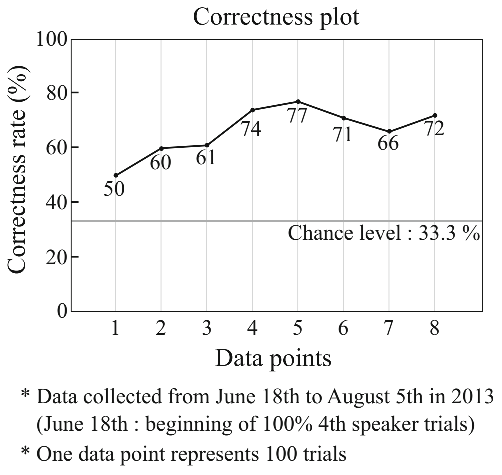


Fig. 2 The cat’s discrimination of three sounds played from a single loudspeaker

interacted with the touchscreen and feeding mechanism appropriately, and learned to touch the target image after hearing a sound.

Requirements

Required technical skills

Some skill in programming and interfacing to a microcontroller is required to initially set CATOS up for a new animal species to test. Beginner or intermediate level of Python scripting is required depending on the complexity of experimental trials, however, Python is a very intuitive language and easy to learn (Lindstrom, 2005; Fangohr, 2004). Also, handling the microcontroller necessitates only a minor work load since the microcontroller used in CATOS is Arduino (see http://www.arduino.cc/), which is an open-source microcontroller with many simple instructions and online tutorials from a large user group. Necessary sensors and actuators can be connected to the microcontroller by simply following such tutorials. Thus, while some technical skills and effort are needed for the initial implementation of a system for a particular species and experimental paradigm, these requirements are scaffolded by a large and enthusiastic "maker" community, with on-line question-and-answer forums, magazines, and conferences. Furthermore, once the physical system is built and implemented, any Masters or PhD student with a modicum of programming skill can modify the code to implement particular experiments (we are also developing a GUI-based experiment builder for our system requiring no programming at all, not described here).

Software & hardware

The CATOS system can potentially run on any commercially available computer. The main programming language used is Python (Version 2.7.3, retrieved in November 2012, from http://www.python.org/), and CATOS uses many external packages such as OpenCV (Bradski, 2000), NumPy/SciPy (Jones et al., 2001), and Matplotlib (Hunter, 2007), which are also open-source libraries, see Table 1.

The current sensory input processing of CATOS is based on computer vision algorithms, implemented using the open-source OpenCV library (version 2.3.1a, retrieved on February 12, 2012, from http://www.opencv.org/), and several sensors via a microcontroller. Also, a touchscreen with Surface Acoustic Wave touch technology was integrated. The visual observational features of CATOS are currently based on motion detection. This was accomplished using commercially available webcams and OpenCV, which is a library for real-time computer vision that including over 500 functions (Bradski and Kaehler, 2008). By using tools in the OpenCV library, CATOS users can easily expand the computer vision algorithms, depending on the purpose of a project. The wide variety of tools implemented in OpenCV means that a vast variety of video analysis routines are available to our system. These include algorithms to detect and recognize faces, identify objects, classify actions in videos, track camera movements, identify movement and track moving objects, or follow eye movements. The current implementation uses only a small fraction of these capabilities.

Table 1 Summary of CATOS system dependencies: all open-source

Software	(Tested) Version	Source
python	2.7	python.org
OpenCV	2.3 & 2.4	opencv.org
ffmpg	1.1	ffmpeg.org
pyaudio	0.2	people.csail.mit.edu/ hubert/pyaudio
numpy	1.6	scipy.org
scipy	0.11	scipy.org
wxPython	2.9	wxpython.org
pyserial	2.6	pyserial.sourceforge.net
arduino	1.0	arduino.cc
matplotlib	1.2	matplotlib.org

The interface between the computer and the various sensors/actuators was implemented using the open-system microcontroller Arduino. Arduino is an inexpensive open-source electronics-prototyping platform with a large and growing user community, which has a wide variety of extension systems termed 'shields' usable for different purposes, such as sensing temperature and controlling various types of motors.

In summary, we believe that by integrating the open toolkits into a single framework for animal behavioral experiments, we leverage both the low price and the great power and flexibility embodied in the new open-source/maker movement. While this requires a certain degree of technical expertise to implement a system, and is thus not appropriate for everyone, this free and flexible system yields considerable advantages over commercial or closed-source systems, not only due to the low price but because the large user community means these open systems will be well maintained, regularly updated, and will grow over time.

Functional description of CATOS

The overall system consists of a combination of software and hardware components; for a schematic overview, see Fig. 3. The main software package is a multi-threaded

Python script, called 'AA' (Agent for Animal), which runs all of the necessary processes in CATOS and communicates with the microcontroller program. The microcontroller program operates sensors and actuators based on communication with 'AA'. The hardware components are composed of multiple devices, some of which are directly connected to the computer via USB cables, others via electric connections to the general-purpose input/output pins of the microcontroller. The microcontroller itself is connected to the computer via a USB cable. The visual and auditory hardware devices are controlled by modules of the 'AA' program using external libraries. Other devices, connected to the general-purpose input/output pins of the microcontroller, are controlled by the microcontroller program. The 'AA' program communicates with the microcontroller program via a serial protocol connection for sending commands to actuators and receiving values from sensors, using the PySerial library.

Software in the overall structure

Once 'AA', the main software module of this system (Fig. 3) starts, it continuously runs several processes in parallel, using the 'multiprocessing' package of Python, until the user terminates the program. The number of processes can be changed and can be turned on or off depending on the requirements of the experiment. These processes include a video-in process for each camera, a video-out process, an audio-in process, an audio-out process, a schema process (see descriptions below), and a message-board process. Even though some of these processes are quite simple, they were implemented as separate processes to prevent them from interfering with each other and/or becoming a processing bottleneck. The system has to process the visual, auditory, and other sensory and motor information simultaneously and in real time to recognize changes in the environment and respond to them appropriately, which can be computationally demanding. We also considered 'threading' rather than 'multiprocessing', but CPU-bound tasks such as heavy calculations on images and/or sounds were more of an issue than I/O bound tasks.

The output data (video input images, recorded WAV files, movement records, CSV files for trial results, and the log file), are temporarily stored in an 'output' folder. After the end of each daily session, all of these output files go through an "archiving" process that can include, but is not restricted to, generating videos, generating images with movement markings, and moving different types of files into the categorized subfolders of a timestamped archive folder.

The following subsections describe the logical flow of important processes implemented in the current system. These processes can be broadly modified depending on the specific training and/or experiment requirements.

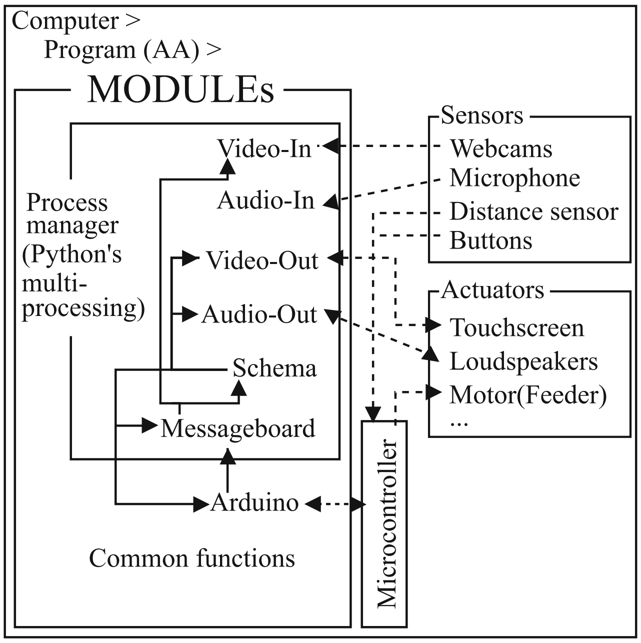


Fig. 3 Schematic diagram of the overall system. *Solid lines* denote the messages exchanged between the modules. *Dotted lines* denote the control connections between the software and the hardware components

Main loop

The main loop procedure launches all necessary processes and then simply waits for user input to terminate the program.

Schema process

The schema process determines the overall behavior of the system, including the training or testing procedures. Specifically, a training or testing procedure in this process can differ considerably depending on the study species and experimental design. The current schema process is an instance used in the first test system with a cat, in which the cat had to press the correct button after a sound stimulus was played through the loudspeakers (see section 'Testing CATOS with a cat').

Video-in process

The video-in process is used not only for recording the current scene, but also for recognizing movements, extracting moving objects (background subtraction) and recording the calculated movement data. The sequence of images is obtained continuously from two webcams, positioned about

3 m above the device, on the wall of the experimental room. Once a motion is detected, the program starts to store images in JPEG format to a temporary folder until no more motion is detected for 3 s. It also stores the records of movements such as the center point of movements and center points of foreground blobs. The stored series of JPEG images in the temporary folder are compressed into one MP4 video file using FFmpeg (http://www.ffmpeg.org) when the daily session is finished.

The most crucial step used for recognizing an animal is motion detection. This algorithm searches for any change in a recently obtained series of images and returns the center point of overall change. Four functions from OpenCV library were used to obtain this information: cv.RunningAvg, cv.AbsDiff, cv.Canny, and cv.FindContours. (Bradski & Kaehler, 2008) An example of motion detection is illustrated in Fig. 4.

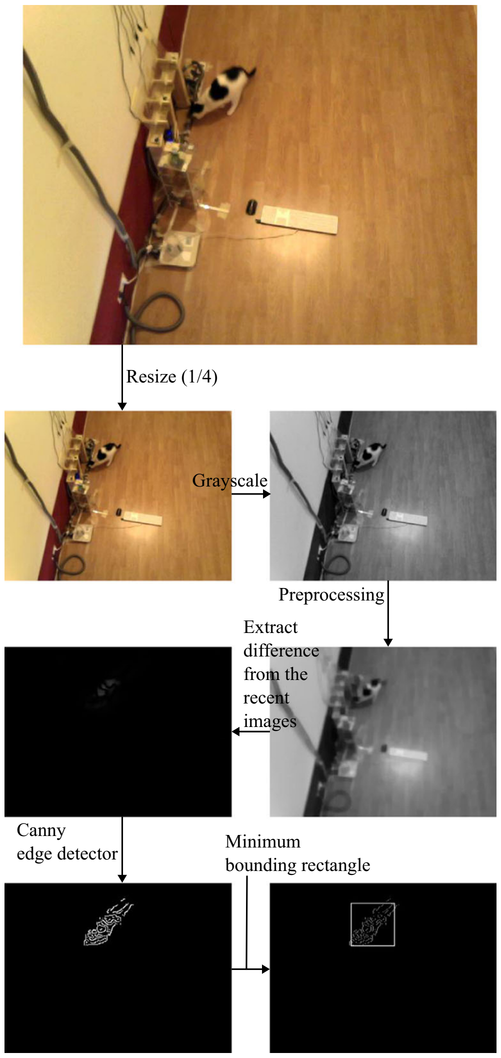

Resize (1/4)
Grayscale
Preprocessing
Extract difference from the recent images
Canny edge detector
Minimum bounding rectangle

Fig. 4 Example of motion detection processing

Video-out process

The video-out process displays images on the touchscreen and processes touches to the screen during a trial. It uses the wxPython Graphical User Interface toolkit (retrieved from http://www.wxpython.org in March, 2013). This process allowed an animal subject, a rat, to respond during a trial by touching the screen in our second test scenario.

Audio-in process

The audio-in process can be used to record any sound around the microphone. Similar to the video-in process, when the root-mean-square amplitude goes above a user-defined threshold, sound recording begins, and the sound is recorded into a WAV file until the amplitude drops under the specified threshold for 3 s. In this instance, a high-pass filter was applied before the measurement of RMS, to prevent the inappropriate recording of a door closing or other incidental noises in the vicinity of the experimental space.

Audio-out process

The audio-out process plays auditory stimuli (WAV files) through the loudspeaker(s).

Feeder

The feeder (Fig. 5) used in this study is a custom-designed device, which consists of an Arduino microcontroller; (http://www.arduino.cc/), a motor shield for the microcontroller, a servomotor, and a frame encasing the whole feeder. The feeder works by rotating the servomotor by a certain number of degrees so that dry food pellets can be dispensed to the animal after a correct response.

Physical interface using a microcontroller

Communication between the Arduino chip and the main computer is accomplished with the Arduino module of the 'AA' program using 'pySerial'. A diagram of wiring components can be found at https://github.com/jinook0707/CATOS_alpha.

The main functions of this subsystem for the cat are: (a) three mechanical buttons providing the response interface, (b) an IR distance sensor that measures the amount of food left, (c) a servomotor which dispenses the food by briefly opening the bottom of the food container.

The main functions of this subsystem for the rat are: (a) a temperature sensor measuring the temperature inside the protective wooden enclosure, (b) two fans which are turned on when the temperature sensor indicates the temperature is too high, (c) a photocell sensor measuring the ambient light level, (d) a light bulb which can be turned on when the photocell sensor indicates the ambient light level is below a user-defined threshold, (e) a servomotor which dispenses food by turning the Archimedes' screw back and forth, (f) a piezoelectric sensor which is polled during servomotor actuation, in order to confirm that the food reward was successfully dispensed.

The utility program: AA_Data viewer

In addition to the system described so far, we also implemented a Python program for analyzing the recorded data using the wxPython GUI toolkit ('AA_DataViewer'). This program allows rapid inspection of overall performance, and a single-image condensed overview of the video footage of any chosen trials (see Fig. 6). AA_DataViewer loads the log file, the result file (CSV format) containing the results of the trials, the movement record CSV files, the MP4 video files, and any recorded WAV files from a timestamped folder containing all data collected for one daily session. The JPEG image in Fig. 6 shows the blobs representing the movements of the cat. The circles represent the positions of the blobs and the color of a circle at the beginning of the recorded video is black, then progressively becomes lighter as the video proceeds. Lines connecting multiple circles mean that those blobs occurred at the same time. AA_DataViewer can also display a graph showing subject performance over selected sessions. The 'archive' folder contains sub-folders, each of which in turn contains all the data for a session. When the 'select sessions' button is clicked, a pop-up window appears for selecting multiple folders. The result data from these selected sub-folders is plotted using the Python graphing library Matplotlib. By visualizing the data for a chosen period, it helps the trainer or experimenter to quickly assess training progress.

Implementation effort

Initial creation of a CATOS system requires someone with basic knowledge of Python programming, plus experience in building hardware. The main software used for running the first testing with a cat is available at https://github.com/jinook0707/CATOS_alpha, as open-source code under GNU General Public License, version 3. This provides a framework for expansion, in which all difficult programming challenges (e.g., multi-processing, message passing, serial communications, and data archiving) have already been solved. The web site also provides plans for hardware, electrical circuit diagrams, flow charts of software, and other useful information.

Each software package listed in Table 1 was installed separately, however, most of them provided package installation files, therefore, the installation did not consume a significant amount of time. There are also Python systems, like Enthought's Canopy (free to academics), that can install and manage all these libraries with its graphical user interface. Before animal training, an initial CATOS physical setup for a new species was conducted. This initial setup took several weeks, mainly due to building a hardware frame for protecting animals and the device. The software setup required Python scripting in the schema, audio-out, and video-out modules. The number of code lines that changed

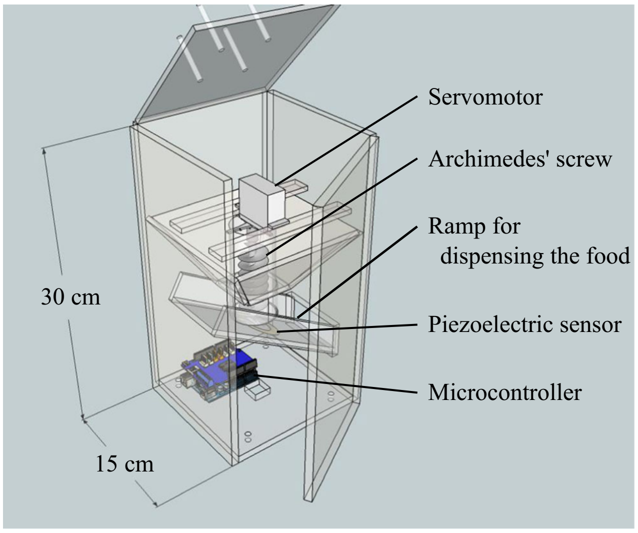
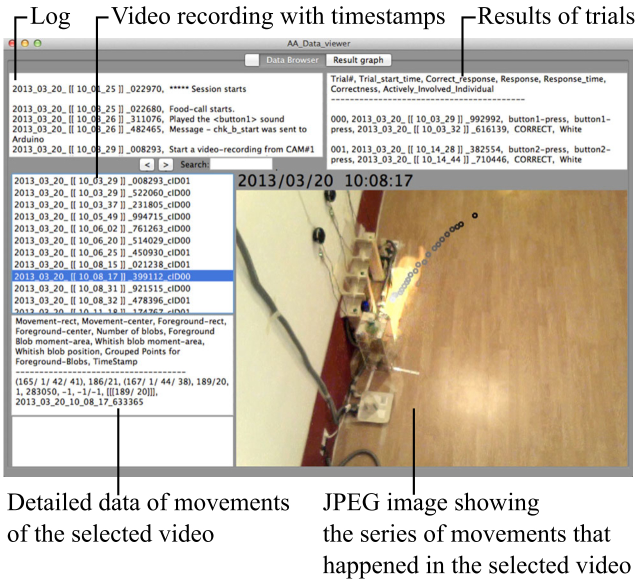


Fig. 5 Custom feeder used in this study


Log
Video recording with timestamps
Results of trials
AA_Data_viewer
Data Browser
Result graph
< >
Search:
Detailed data of movements of the selected video
JPEG image showing the series of movements that happened in the selected video

```
2013_03_20_ [[ 10_01_25 ]] _022970,  ***** Session starts

2013_03_20_ [[ 10_03_25 ]] _022680,  Food-call starts.
2013_03_20_ [[ 10_03_26 ]] _311076,  Played the <button1> sound
2013_03_20_ [[ 10_03_26 ]] _482465,  Message - chk_b_start was sent to
Arduino
2013_03_20_ [[ 10_03_29 ]] _008293,  Start a video-recording from CAM#1
```

```
Trial#, Trial_start_time, Correct_response, Response, Response_time,
Correctness, Actively_Involved_Individual
-----------------------------------------

000, 2013_03_20_ [[ 10_03_29 ]] _992992,  button1-press,  button1-
press, 2013_03_20_ [[ 10_03_32 ]] _616139,  CORRECT,  White

001, 2013_03_20_ [[ 10_14_28 ]] _382554,  button2-press,  button2-
press, 2013_03_20_ [[ 10_14_44 ]] _710446,  CORRECT,  White
```

```
2013_03_20_ [[ 10_03_29 ]] _008293_cID01
2013_03_20_ [[ 10_03_29 ]] _522060_cID00
2013_03_20_ [[ 10_03_37 ]] _231805_cID00
2013_03_20_ [[ 10_05_49 ]] _994715_cID00
2013_03_20_ [[ 10_06_02 ]] _761263_cID00
2013_03_20_ [[ 10_06_20 ]] _514029_cID00
2013_03_20_ [[ 10_06_25 ]] _450930_cID01
2013_03_20_ [[ 10_08_15 ]] _021238_cID01
2013_03_20_ [[ 10_08_17 ]] _399112_cID00
2013_03_20_ [[ 10_08_31 ]] _921515_cID00
2013_03_20_ [[ 10_08_32 ]] _478396_cID01
```

```
2013/03/20  10:08:17
```

```
Movement-rect, Movement-center, Foreground-rect,
Foreground-center, Number of blobs, Foreground
Blob moment-area, Whitish blob moment-area,
Whitish blob position, Grouped Points for
Foreground-Blobs, TimeStamp
-----------------------------------
(165/ 1/ 42/ 41), 186/21, (167/ 1/ 44/ 38), 189/20,
1, 283050, -1, -1/-1, [[[189/ 20]]],
2013_03_20_10_08_17_633365
```

Fig. 6 Screenshot of AA_DataViewer; Browsing session data

for switching from one test to another was 200–300. This could take 1 or 2 days to 1 week for a beginner-level Python programmer depending on the complexity of the desired trial procedure.

Once the initial system has been created and set up, the effort required to run an experiment becomes minimal. For example in our cat training experiment, the experimenter assessed acquired videos and result CSV data daily. This maintenance was conducted for about 15–30 min each day, coupled with regular animal-keeping tasks such as changing water and cleaning.

Observable advantages of the current CATOS version

By using CATOS, we significantly reduced data storage and human resources for running animal experiments. Two webcams observed the experimental area for 8–12 h per day for about 10 months, and the entire output data during this period, including movement records, MP4 video files, and JPEG image files, only took 50.6 gigabytes of storage. To obtain a rough idea of the degree of reduction in data storage that was achieved using the system, we assessed the number of recorded frames in the video recording using data from 15 randomly chosen days, involving a total observation period of 112.8 h. The number of frames recorded during this period was only 206,024, with an average FPS (frames per second) of 7.5. Therefore the video recordings of approximately 27,470 s (=7.6 h) were stored, which is about 6.7 % of the entire observation period. These specific numbers are not definitive since they can fluctuate greatly depending on the increase or decrease of the subject's movements as well as the system settings, but the point is that most of the useless video recordings were filtered out by this system using efficient sensor information processing. Also, human presence was not necessary during a training or testing session, as already described. The creation and modification of the system modules, data transfer, and other maintenance work obviously required some human interaction, but no human time and effort was required to execute the training and testing sessions. Although the sessions themselves did not require a human to be present, the animal's performance with the system had to be periodically assessed and analyzed. How much food the animals obtained, or how many correct and incorrect trials occurred, could be quickly assessed from the result file of a session, which displayed the number of correct and incorrect trials with timestamps. Table 2 gives an example of such a result file. A more detailed assessment was also possible, involving watching the recorded videos, which takes more time, but as we showed, the total amount of data was greatly reduced. The data viewer utility program displayed all the timestamps and its condensed JPEG image, which presented a brief summary of the movement detected in the recorded video clip shown in Fig. 6. Thus, simply browsing the JPEG images was often enough to assess a session. Only when it was necessary, was a more detailed assessment performed by playing the video recordings of trials.

Discussion

CATOS is currently in its beta stage. Although the system is stable, more tests with further species and environments are necessary. The potential of CATOS to be used in training and testing animal cognition was demonstrated with two animal species and we believe CATOS also shows considerable promise for saving human resources.

CATOS has several advantages. (a) CATOS itself and its dependencies are all open-source projects, therefore, there are no pre-defined limits to how the system could be used and adjusted for future research purposes. Its free open-source software and common inexpensive hardware parts make CATOS a flexible and low cost system, increasing its applicability to a variety of species, tasks, and locations.

Table 2 Example of the result file in CSV format

Trial#	Trial_start_time	Correct_response	Response	Response_time	Correctness
000	2013_03_17_10_54_13_965895	button2-press	button3-press	2013_03_17_10_54_15_388055	INCORRECT
001	2013_03_17_11_12_10_886611	button3-press	button3-press	2013_03_17_11_12_15_867220	CORRECT
002	2013_03_17_11_29_12_784309	button2-press	button2-press	2013_03_17_11_29_16_663150	CORRECT
.	.	.	.	.	.
.	.	.	.	.	.
.	.	.	.	.	.
017	2013_03_17_17_22_17_300355	button3-press	button2-press	2013_03_17_17_22_22_520951	INCORRECT
018	2013_03_17_19_37_46_166803	button1-press	TIMEOUT	2013_03_17_19_38_15_833813	TIMEOUT

(b) The amount of data storage and human resources required to train animals and analyze the stored data is substantially reduced when CATOS is employed, compared to conventional systems for animal training and testing. (c) CATOS can be freely accessed by animals at any time without any physical constraints on animals. Also, the concept of gamification was introduced in the animal training process with a different level of difficulties in each training phase. These features enriched the animals' environments.

The most obvious limitation of using an open-source system such as CATOS is that the user has to possess a certain degree of knowledge across different fields such as computer science, engineering, and electronics. However, this difficulty has decreased thanks to the recent software and hardware development platforms. CATOS uses Python and its external libraries, and open-source hardware, Arduino. Both are popular in scientific communities and offer intuitive syntax and interface (Lindstrom, 2005; Fangohr, 2004; Pearce, 2012), therefore, one can improve the system with a relatively shallow learning curve. Another potential limitation could be the computational performance in terms of speed. However, the speed of the current version of CATOS was adequate for most purposes in animal behavioral studies. The frames per second of the video module was about 25, when the system was monitoring a scene with a webcam. Although it is often said that script languages gain productivity in exchange for performance, the difference in run time is often tolerable, depending on task and coding style (Prechelt, 2000). Also, there are ways to use compiled C code in Python only for specific computational tasks to exploit the performance of C. (e.g., in the OpenCV computer vision routines)

After an initial 2-month period of implementation, our system was highly reliable, with only a single failure occurring during the 8-month training/testing period (caused by a communication error between the computer and microcontroller, which has been corrected). However, different experimental designs, test species, or research environments might lead to different reliability profiles.

Much of the functionality of the current version of CATOS could alternatively be achieved by using combinations of several commercial products, or building on previously published work in the field, which also incorporates commercial products. The aim of our CATOS project was not to provide a complete all-purpose product working out of the box, but rather to provide an open and flexible system for testing animal behaviors, which is completely independent of any proprietary products and adjustable with low cost. Thus, CATOS fills a previously empty niche in computer-aided research in animal testing.

Further work

In the future, the CATOS framework will be developed further considering the following aspects: (a) Restricting an animal's access to a ROI (region of interest) zone: Only one animal would be allowed into an ROI zone, with no forceful separation from its own residence or social group. Such a feature will help to achieve individual identification, different tasks, and rewards depending on the individual, and so forth. Implementation of access restriction would have to be introduced gradually, so that the individual already in the ROI could leave the zone at any time, but others outside of the ROI are prohibited from entering the zone. (b) Individual identification: If the goal is to have the system respond differently to each individual, it will be necessary for the system to recognize individuals automatically. RFID (radio frequency identification) technology or a computer vision algorithm exploiting a specific color or shape are both currently under consideration. Although RFID technologies are appropriate for some tasks, and offer very reliable identification when they work (see, e.g., Fagot and Bonté 2010), the short range of data transfer means that the ID chip on (or in) the animal subject needs to come very close to the chip reader. This is not always practical, nor would injecting chips into some animal subjects be desirable (e.g., wild animals). Either technology, or both, might be used in the future depending on the type of animals. (c) Slave units through a wireless network: The observation feature of the current system can cover a small area such as the room used in this study, but if multiple slave units with microcontrollers or single-board computers, which can send information to a master computer, are implemented, the scope of the system could be expanded to cover a larger and more complex area. Such a multi-unit wireless system could prove particularly useful for animal observation tasks.

Author note This work was funded by the European Research Council Advanced Grant SOMACCA, Nr. 230604 to WTF.

References

Bellotto, N., Sommerlade, E., Benfold, B., Bibby, C., Reid, I., Roth, D., Fernandez, C., Gool, L.V., & Gonzalez, J. (2009). A distributed camera system for multi-resolution surveillance. In *Proceedings of the 3rd ACM/IEEE Int. Conf. on Distributed Smart Cameras (ICDSC) 2009* (pp. 1–8): IEEE.

Bradski, G. (2000). The OpenCV Library. *Dr. Dobb's Journal of Software Tools*, *25*(11), 122–125.

Bradski, G., & Kaehler, A. (2008). *Learning OpenCV: Computer vision with the OpenCV library*: O'Reilly.

Deterding, S., Dixon, D., Khaled, R., & Nacke, L. (2011). From game design elements to gamefulness: defining "gamification". In *Proceedings of the 15th International Academic MindTrek Conference: Envisioning Future Media Environments*.

Fagot, J., & Bonté, E. (2010). Automated testing of cognitive performance in monkeys: Use of a battery of computerized test systems by a troop of semi-free-ranging baboons (*Papio papio*). *Behavior Research Methods*, *42*(2), 507–516.

Fagot, J., & Paleressompoulle, D. (2009). Automatic testing of cognitive performance in baboons maintained in social groups. *Behavior Research Methods*, *41*(2), 396–404.

Fangohr, H. (2004). A comparison of C, Matlab, and Python as teaching languages in engineering. In *Computational Science-ICCS 2004* (pp. 1210–1217): Springer.

Huber, L., Heise, N., Zeman, C., & Palmers, C. (2015). The ALDB box: Automatic testing of cognitive performance in groups of aviary-housed pigeons. *Behavior Research Methods*, *47*(1), 162–171.

Hunter, J.D. (2007). Matplotlib: A 2D graphics environment. *Computing In Science & Engineering*, *9*(3), 90–95.

Jones, E., Oliphant, T., & Peterson, P. (2001). SciPy: Open-source scientific tools for Python.

Kangas, B.D., & Bergman, J. (2012). A novel touch-sensitive apparatus for behavioral studies in unrestrained squirrel monkeys. *Journal of Neuroscience Methods*.

Kritzler, M., Jabs, S., Kegel, P., & Krüger, A. (2008). Indoor tracking of laboratory mice via an RFID-tracking framework. In *Proceedings of the first ACM international workshop on mobile entity localization and tracking in GPS-less environments* (pp. 25–30): ACM.

Lindstrom, G. (2005). Programming with Python. *IT Professional*, *7*(5), 10–16.

Markham, M.R., Butt, A.E., & Dougher, M.J. (1996). A computer touch-screen apparatus for training visual discriminations in rats. *Journal of the Experimental Analysis of Behavior*, *65*(1), 173–182.

Pearce, J.M. (2012). Building research equipment with free, open-source hardware. *Science*, *337*(6100), 1303–1304.

Prechelt, L. (2000). An empirical comparison of C, C++, Java, Perl, Python, Rexx and Tcl. *IEEE Computer*, *33*(10), 23–29.

Roberts, C.M. (2006). Radio frequency identification (RFID). *Computers & Security*, *25*(1), 18–26.

Russell, S., & Norvig, P. (2009). *Artificial Intelligence: A Modern Approach*, 3rd edn: Prentice Hall.

Steurer, M.M., Aust, U., & Huber, L. (2012). The Vienna Comparative Cognition Technology (VCCT): an innovative operant conditioning system for various species and experimental procedures. *Behavior Research Methods*, *44*(4), 909–918.

Takemoto, A., Izumi, A., Miwa, M., & Nakamura, K. (2011). Development of a compact and general-purpose experimental apparatus with a touch-sensitive screen for use in evaluating cognitive functions in common marmosets. *Journal of Neuroscience Methods*, *199*(1), 82–86.

Vallejo, D., Albusac, J., Jimenez, L., Gonzalez, C., & Moreno, J. (2009). A cognitive surveillance system for detecting incorrect traffic behaviors. *Expert Systems with Applications*, *36*(7), 10503–10511.