A Low-Cost Platform for Eye-Tracking Research: Using Pupil© in Behavior Analysis

Journal:	*Journal of the Experimental Analysis of Behavior*
Manuscript ID	JEAB-2017-0127.R3
Manuscript Type:	Technical Article
Date Submitted by the Author:	06-Apr-2018
Complete List of Authors:	Picanço, Carlos; Universidade Federal do Pará Tonneau, François; Universidade Federal do Pará
Keywords:	eye tracking, discrimination, conditioned reinforcement, open-source software, humans

Abstract: Tracking eye movements is being increasingly recognized as a valuable source of information about stimulus control. So far, however, eye-tracking research has suffered from accessibility issues, with expensive hardware and closed-source software. In this article we review Pupil©, an eye-tracking platform developed by Pupil Labs and that combines open-source software with low-cost hardware components. We offer concrete recommendations about Pupil use in stimulus-control research and we show how the software can be extended to automatize the analysis of gaze data. Finally, we present the results of a study of visual discrimination and conditioned reinforcement conducted with Pupil, establishing the usefulness of this platform as a research tool in behavior analysis.

A Low-Cost Platform for Eye-Tracking Research: Using Pupil© in Behavior Analysis

Carlos R. Picanço

François Tonneau

Author Note

Carlos R. Picanço, Center for Theory and Research on Behavior, NTPC – Federal University of Pará, Belém, Brazil. François Tonneau, Center for Theory and Research on Behavior, NTPC – Federal University of Pará, Belém, Brazil.

Correspondence may be sent to either author at Universidade Federal do Pará, Núcleo de Teoria e Pesquisa do Comportamento, Rua Augusto Corrêa, 01, Guamá, Belém PA 66075-110, Brasil. E-mail: cpicanco42@gmail.com; francois.tonneau@gmail.com

Abstract

Tracking eye movements is being increasingly recognized as a valuable source of information about stimulus control. So far, however, eye-tracking research has suffered from accessibility issues, with expensive hardware and closed-source software. In this article we review Pupil©, an eye-tracking platform developed by Pupil Labs and that combines open-source software with low-cost hardware components. We offer concrete recommendations about Pupil use in stimulus-control research and we show how the software can be extended to automatize the analysis of gaze data. Finally, we present the results of a study of visual discrimination and conditioned reinforcement conducted with Pupil, establishing the usefulness of this platform as a research tool in behavior analysis.

Keywords: eye tracking, observing, discrimination, conditioned reinforcement, open-source software, humans

A Low-Cost Platform for Eye-Tracking Research: Using Pupil© in Behavior Analysis

Most of the activities recorded in the behavior-analytic laboratory consist of motor responses directed at devices such as levers, buttons, or keys. Studies of stimulus control in humans, for example, often involve matching-to-sample tasks in which people press buttons on a keyboard or move a computer mouse. Although much valuable information about stimulus control has been gathered by these methods, more recently behavior analysts have started to diversify their dependent variables and look for additional information beyond the recording of device-directed motor responses (cf. Palmer, 2010). In visual discrimination tasks, for example, recording the subjects' gaze along their button-pressing responses has already shown promise as a valuable source of information (e.g., Dube et al., 1999, 2010; Huziwara, de Souza, & Tomanari, 2016; Perez, Endemann, Pessôa, & Tomanari, 2015; Steingrimsdottir & Arntzen, 2016). This line of research has also rekindled behavior analysts' more general interest in the study of eye movements (e.g., Paeye & Madelain, 2011), a topic that Holland and colleagues (Holland, 1958; Schroeder & Holland, 1968, 1969) had previously pioneered.

Regardless of their focus of study (understanding stimulus control or eye movements per se), the behavior analysts interested in pursuing eye-movement research have been facing two hurdles. The first one is cost, as the major commercial platforms for eye-tracking research have been so far exceedingly expensive. The second hurdle stems from the closed-source approach to software that commercial platforms tend to adopt. Closed-source software cannot be freely modified and its limitations are therefore difficult to overcome. For example, if the output provided by the software is a video of the stimulus array with a visual marker to indicate gaze location, then the video must be codified by an observer to identify which stimuli the subjects looked at during the session (e.g, Perez et al., 2015). Obtaining inter-observer agreement requires at least another round of codification and is even more time-consuming.

In this article we discuss Pupil©, a platform for eye-movement research created by Pupil Labs that overcomes these limitations. The web site of the company is currently located at https://pupil-labs.com/. Pupil has been an open-source project since 2012, and the surrounding community has been steadily growing ever since (see https://pupil-labs.com/chat). The Pupil platform comprises hardware and software components. The hardware is available at a fraction of the cost of other commercial alternatives, and the software is licensed under the Free Software Foundation's LGPLv3. Not only is Pupil's source code guaranteed to be forever available (as a LGPLv3 license cannot be revoked), it can be easily extended by writing plugins in the Python language. We show below that these features make of Pupil an attractive option for eye-tracking research in behavior analysis.

We start by describing Pupil's workings just in enough detail to make the operation of the platform easily understood. Interested readers will find fuller specifications of the hardware and software components in the technical article by Kassner, Patera, and Bulling (2014) and in Pupil's online documentation. We discuss the technical difficulties we encountered in running the Pupil platform and we explain how they can be resolved. Finally, we present a simple study of stimulus discrimination in humans that demonstrates the feasibility of using Pupil in behavior-analytic research.

The Pupil Platform

How Pupil works

Pupil's eye-recording mechanism involves two cameras, an *eye* camera directed at the subject's eye, and a *world* camera focusing on the "world" or scene in front of the subject. Thus, in a typical gaze-recording study, the scene comprises the stimuli that the experimenter presents and that the subject is supposed to look at. The Pupil software analyzes each frame from the eye camera, applying sophisticated contour-detection algorithms to detect pupil shape and compute its orientation with respect to the orbit of the eye (Kassner et al., 2014). Meanwhile, the software registers the content of the scene from the output of the world camera.

Now, knowing the orientation of the eye relative to its orbit does not translate automatically into knowing where the subject is looking in terms of scene coordinates. In order to move from the former to the latter, Pupil must be taught the correspondence between pupil orientations (as detected by the eye camera) and coordinates of the scene (as detected by the world camera). This is the purpose of *calibration*, a procedure which must be carried out on each subject before each recording session starts. During calibration, the subject is asked to look at different reference points, the positions of which are known beforehand to Pupil (see below for more details). From its recordings of pupil orientations and knowledge of the coordinates of reference points in the scene, Pupil learns the mapping from the former to the latter. (More technically: The coefficients of a bivariate polynomial are adjusted so as to predict x, y reference-point coordinates from pupil orientations. Once calibration is over, this polynomial is used during the session to translate detected pupil orientations into estimated gaze locations. Notice that this polynomial is subject-specific, depending as it does on each person's facial morphology.)

Hardware

The computer needed to run the Pupil platform must be equipped with (at least) a Core i3 processor and two USB ports. We recommend using USB ports with two separate controllers, as a single controller may not be able to cope with the required bandwith. Aside from the computer, the hardware needed to run Pupil consists minimally of (a) digital cameras, (b) a headset on which the cameras should be mounted, (c) USB cables to connect the cameras to the computer, and (d) a clip to secure the cables and keep them from pulling the headset down. The hardware is modular and exists in different versions, depending for instance on whether pupil detection is supposed to be monocular or binocular. Regardless of the chosen version, the full setup can be bought from Pupil Labs; alternatively, any of the components from (a) to (d) can be bought from other providers and assembled into a functioning system by following Pupil's do-it-yourself guide (currently located at https://docs.pupil-labs.com). Thus, to carry out the present study, the only component we bought from Pupil Labs was the headset in its monocular version.[1] Apart from the headset, we bought two UVC-compliant cameras[2], two USB cables, and a cable clip from retail stores.

Our eye and world cameras were a LifeCam Microsoft® HD-6000 and a Logitech Webcam C615, respectively, both with a 30-Hz frame rate and a 1280 × 720 pixel count. Although these proved adequate for our study, we recommend using an eye camera with a higher frame rate (as the latter increases the frequency of frames with a detected pupil contour: see **Optimizing Data Quality** below) and a world camera with a larger pixel count (as the latter makes it easier to apply contour-detection algorithms to scene contents: see **Data-Analysis Automation**). Following Pupil Labs' do-it-yourself instructions, we removed the cameras from their casings and we replaced the LED of the eye camera by an SFH-4050 IR-LED bought separately. (People's eyes must be illuminated by an infrared LED to be detected by the Pupil software.) Finally, we mounted the cameras on the headset and connected them to the USB cables.

Figure 1 will give the reader a good idea of what the gaze-recording setup looked like at the time of the study. With the exception of a more recent eye camera (an ELP-USBFHD01M-L21 model), the hardware shown in the figure is actually identical to that of our pilot study. The world camera on the subject's forehead faces the reader, whereas the eye camera faces backward toward the subject. The white tape at the bottom of the world camera helps in maintaining the quality of world-camera detection by deflecting illumination from the eye-camera LED. The figure also shows the USB cables clipped to the subject's clothing as well as the custom chin rest we employ to stabilize the subject's head. The projector we use to display stimuli is visible in the background.

Software

The Pupil software is cross-platform (GNU/Linux, OSX, Windows) and available free of charge from Pupil Labs. It comprises two separate programs, *Capture* and *Player*. Both programs present a graphical user interface and include a point-and-click control panel that allows users to configure the software and run a variety of procedures.

Aside from the control panel, the Capture program displays two windows, visible side by side at all times during the session. One of the windows shows a current snapshot of the subject's eye, as seen by the eye camera. Whenever the subject's pupil is detected, it appears with a red contour and a dot at its center. The other window just shows the current output of the world camera. A first function of Capture is to allow the experimenter to check at a glance that pupil detection works correctly, as can be judged from the red contour and the presence of the dot in the eye. A second function of Capture is to calibrate the eye-recording system. As explained above, calibration is fundamental because it establishes the correspondence between pupil orientations and scene coordinates. Capture permits a variety of calibration methods, based on ad-hoc markers or other features present in the scene.

The third function of Capture is to record gaze data during the session. When analyzing each video frame from the eye camera, Capture segments the image to isolate a candidate for the contour of the pupil and fits an ellipse to the detected contour. Along with the position of the ellipse, Pupil computes a goodness-of-fit measure between 0 and 1 that serves as a degree of confidence in the correctness of pupil detection (for technical details see Kassner et al., 2014). Only the records with a degree of confidence higher than a given threshold are saved for later use. The value of this threshold can be configured from Capture's control panel. Using a threshold of 1, for example, will guarantee a high degree of accuracy but entail the rejection of at least some video frames, whereas a threshold of 0 will exclude nothing but guarantee no accuracy in detection. In practice, we have found that a confidence threshold of .80 strikes a good balance between accuracy and inclusiveness.

The data recorded by Capture can be fed after the recording session into Player, Pupil's other main program. The first function of Player is to replay the video and combine it with gaze data (as estimated from recorded eye orientations) to allow the experimenter to review how the session went. Different options are available to represent gaze data pictorially. Gaze locations in individual frames can be represented by a cross or a circle, for example, or the subject's whole scan path can be superimposed on the video. The other main function of Player is to transform gaze data and export them in different formats. Python plugins can be used at this stage to perform additional data analyses before exportation (see below).

Data-Analysis Automation

The system we have been describing so far is perfectly sufficient to perform eye-tracking research. Each recording session should start by checking that the subject's pupil is detected correctly and by performing system calibration. Eye recording will then be launched from Capture by pressing a hotkey. During the session, stimuli could be projected on a screen by a program independent from Pupil, or they could even be presented manually. Pressing another hotkey will end the session at any time. The experimenter will then be able to replay the session with Player and appraise what the subject was looking at by watching the gaze marker(s) superimposed on each video frame.

A problem with this data-analysis method, of course, is that it requires a human observer to categorize the subject's gaze locations, a procedure we have already criticized as time-consuming. The nature of Pupil's open-source software, however, makes it easy to extend the recording system with one's own data-automation scripts. Moving toward automated, and therefore more efficient, gaze-data analyses requires two additions to the Pupil platform.

First, the stimuli used during the session should be presented by a computer program (manual presentation will not do, unless one chooses to detect the stimuli from their contours on each recorded frame: see below), and this program should save stimulus events with time stamps commensurate with those of Pupil Capture. The simplest way to do so is to have the program access the same system-time function that Pupil employs by default (on GNU/Linux operating systems, for example, this function is *clock_gettime*). The stimulus-presentation program we used in our study, for example, saved stimulus-onset times in a plain-text file along with the normalized screen coordinates of each stimulus being presented.

Second, a plugin must be added to Pupil Player to transform eye-recording data into a form more suitable to data analysis. Remember that the main output from Capture consists of gaze coordinates within the "scene" (basically, a rectangle of pixels that represents the "field of vision" of the world camera). Analyzing gaze data in relation to experimenter-defined stimuli requires knowing where the stimuli are located in the scene. Contour-detection algorithms such as those of OpenCV (an open-source library freely redistributed with Pupil) can be used to this end, but they work best when applied to simple shapes on a high-contrast background. Accordingly, instead of trying to detect individual stimuli, we recommend detecting the four corners of the computer screen and reexpressing gaze coordinates in relation to them. We have found this detection method to be effective, as long as the screen background is white and contrasts with its surroundings and whether subjects look at an image of the computer screen projected on the wall or at the computer screen itself. Our plugin can be found at:
https://github.com/cpicanco/pupil-pugin-and-example-data
along with instructions on how to install the plugin and activate it from within Pupil Player. (The folder also contains session data and R scripts for gaze-data correction and analysis, see below).

Regardless of the chosen approach to stimulus detection, we recommend converting gaze locations to normalized Cartesian coordinates in the plane of the computer screen. We also recommend saving the transformed data in a plain-text, comma- or tab-separated file, one line per frame, one file per session. Data analysis can then proceed by comparing gaze locations on the screen (as estimated by Pupil) with those of the stimuli presented during the session (as reported by the data-presentation program).

Optimizing Data Quality

The data recorded by Pupil face two possible problems. The first concerns the absolute frequency of gaze detection. Moments during which gaze detection is failing manifest themselves as missing lines in the data file, and they are easy to spot by checking the time stamps of successive records. Gaze detection may of course fail for good reasons (if the subject is squinting, coughing, or even blinking), but gaps in the data record may also indicate that the platform is not working correctly. In this case, however, the problem can be minimized by using an eye camera with a higher frame rate. In practice, we have not found this issue to be significant.

The second problem--in our experience, the most serious with Pupil--involves inaccurate estimates of gaze locations. The top panel of Figure 2 illustrates the issue with the data from one of the subjects (P5, first session) in our study. This panel represents the screen of the computer in normalized Cartesian coordinates, with dotted horizontal and vertical lines crossing at the center of the screen. The contours of the stimuli used during the session (a square and a circle, respectively) appear left and right of the screen center. Each dot indicates the location of the subject's gaze in one video frame, as reported by Pupil Capture. (About 1% of the data points fell outside the screen; they are not shown in the panel.) Clearly, there is something wrong with these data. Whereas one would expect most of the dots to fall within the boundaries of the square and the circle, the cloud of dots reported by Pupil seems displaced to the right and slightly below what its location should be.

The inset of the top panel of Figure 2 (a closeup of the central area of the screen) gives a better indication of what happened to the gaze cloud over the course of this session. Each filled circle in the panel shows the location of the center of the cloud, computed over eight successive blocks of 1,000 video frames each. Given a nominal frame rate of 30 video frames per s, each block thus corresponds to a period of about 1,000/30 = 33 s. The cloud center for each block was defined as the arithmetic mean of the 5th, 10th, 15th, 85th, 90th, and 95th quantiles of the *x* and *y* coordinates of the cloud distribution after all off-screen data points had been eliminated. The idea behind this definition was to obtain a measure of central tendency of the cloud that would be robust to outliers while taking into account a broad fraction of the data. Because the stimuli we used were located symmetrically around the center of the screen (Figure 2), the cloud center and the screen center should have been close to each other at all times. Actually, as shown by the arrow in Figure 2 (inset of the top panel), they were close to each other only on the first block of 1,000 frames. Thereafter, the cloud center started to drift, slighty to the left at first, then more and more to the right and below the center of the screen.

Presumably, this kind of drift is a consequence of the slippage of the headset during the session, a slippage that can also occur right after the calibration routine and even when using a chin rest to stabilize the subject's head. In any case, what can be done to address the issue once the session is over and the data have been collected? Various possibilities suggest themselves, depending on the nature of the experiment and the type of stimulus arrays being employed. In a discrete-trial experiment with a fixation point at the start of each trial, for example, one might compute the difference between the fixation point and gaze location at the start of the trial, and subtract this difference to correct all within-trial gaze coordinates. In an experiment with well-separated groups of objects, one might apply automatic clustering techniques such as DBSCAN (Ester, Kriegel, Sander, & Xu, 1996) to classify gaze locations into distinct categories, and perform all further analyses on the basis of these categories. Clustering techniques have the advantage of not requiring the application of ad-hoc corrections to gaze coordinates; however, clustering may fail to separate the data in distinct categories if the stimuli employed are too close to one another.

As an alternative to clustering, a technique we use in our laboratory consists of (a) computing the center of the gaze cloud over blocks of 1,000 frames, as described above, (b) computing the difference between the cloud center and the screen center on each block, and (c) subtracting this difference from all the data in the block. We will refer to this technique as *cloud-center correction.* Applying cloud-center correction to the dots in Figure 2 gives the results shown in the bottom panel of the figure. After correction, most of the gaze locations fall inside the boundaries of the stimuli, as should be the case. We have found cloud-center correction to be an effective way of dealing with problems such as headset slippage. The method should work properly as long as the stimuli used in the recording session are distributed symmetrically around the center of the screen, which should be the case anyway in most operant studies of visual choice.

Figure 3 summarizes the technical details of hardware and software discussed so far with a big-picture view of the situation, from gaze recording to data analysis. The figure makes it clear that in any gaze-recording study with Pupil, both Pupil Capture and the stimulus-presentation program must rely on common system time for time synchronization. However, aside from Pupil Capture, Pupil Player, and the Python plugin we wrote for the latter, the software components in Figure 3 may be written in any computing language. Thus, although we employed Free Pascal/Lazarus for stimulus presentation and R for data correction and analysis (see below), these programming choices are by no means obligatory.

We now turn to a description of our study, in which we assessed the relation between observing and conditioned reinforcement (e.g., Dinsmoor, 1983, 1985) across and within subjects. The study involved three phases. The first was a baseline phase in which a simple motor response, button-pressing, was reinforced by points in the presence of two alternating stimuli. In the second phase, one of the stimuli (S+) remained correlated with reinforcement, whereas the other stimulus (S-) was correlated with extinction. On the basis of previously reported findings and conditioned-reinforcement theory (e.g., Dinsmoor, 1983; Fantino, 1977; Shahan & Cunningham, 2015; Tomanari, 2004), we expected S+ to become a conditioned reinforcer during this phase, and therefore observing of this stimulus to increase. The third phase was a return to baseline.

We originally planned to reinforce button pressing on a variable-interval schedule, but pilot work showed that using intermittent reinforcement in our setting generated superstitious response sequences (cf. Ono, 1987; Wagner & Morris, 1987) that were highly resistant to extinction. This in turn impeded the establishment of any discrimination in the second phase of the experiment. Accordingly, we decided to switch to continuous reinforcement. This made our procedure non-optimal with respect to visual discrimination, as the points delivered after each response in S+ could themselves serve as cues for reinforcement, overshadowing visual stimuli in the process (Mackintosh, 1977). Using continuous reinforcement allowed the experiment to be shorter, however, and it also paid off by revealing unexpected consistencies in our subjects' observing behavior across phases (see below).

Method

Subjects

Eight college students (4 males, 4 females; age range = 18-27 years) agreed to participate in a study about "perception and learning." All students had normal color vision and were majoring in courses other than psychology. The students were debriefed about the purpose of our research at the end of the study.

Apparatus and Settings

The study took place in a 3 m × 5 m room with windows covered with cardboard so that the level of illumination in the room (range = 320-640 lux), provided by a bulb on the ceiling, could be better controlled. A computer running GNU/Linux on an i3 processor with 2GB of RAM controlled the presentation of visual stimuli on the screen (1280 x 768 pixels) through a custom program written in Lazarus (Picanço, 2017). The same program also controlled sound presentation via the computer loudspeakers and registered button-pressing responses, while Pupil Capture was recording the visual scene as well as the subject's gaze.

During the sessions, an NEC-V260R digital projector displayed the contents of the computer screen on the wall. The subject was seated in an office chair 2.40 m in front of the stimulus display, so that the projection of the computer screen on the wall covered 17 and 10 degrees of visual angle in the horizontal and vertical directions, respectively. A modified doorbell connected to the computer served as a response switch that could be activated by pressing on the doorbell button with one's thumb (minimal mass for switch activation = 50 g). A chin rest was mounted on a metallic frame of adjustable height in front of the chair. The subject wore the Pupil headset with his or her head on the chin rest as in Figure 1.

Procedure

The experiment was run as a single session with three phases, each 5-min long. Excepting during instructions and calibration, two geometrical shapes were present at all times on the screen: a square (left of center) and a circle (right of center), as in Figure 2. Each phase involved a multiple schedule with four components. In Components 1 to 4, the colors of the square on the left and of the circle on the right were, respectively: red and green; red and cyan; blue and cyan; blue and green. Thus, the color of the stimulus on the left changed at the start of odd-numbered components, whereas the color of the stimulus on the right changed at the start of even-numbered components. The duration of each component was a uniform random variable ranging from 4 to 12 s. The duration of each phase was fixed at 5 min. Accordingly, the number of components per phase could vary, but equalled at least 36 (= 9 presentations of the 4 components).

Phases varied only with respect to the reinforcement contingencies for button pressing. During the first, baseline phase of the experiment (A), every button press was reinforced by the sound of a cash register. During the second phase (B), button pressing was still reinforced when the square on the left was red (Components 1 and 2), but was extinguished when the square on the left was blue (Components 3 and 4). The third phase of the experiment was a return to baseline (A), with each button press followed by the cash-register sound. Even though, strictly speaking, the blue square took up the S- function only in Phase B, from now on and for convenience we will refer to the red and blue squares in all phases as "S+" and "S-," respectively.

Before each phase, the experimenter calibrated the system with Pupil's method of Natural-Feature Calibration (see https://docs.pupil-labs.com/#calibration). The reference points for calibration were 15 dots displayed as three rows of five black dots on a white background. After the first calibration and before the first phase, the following instructions (here translated from Portuguese) appeared on the screen:

> Hello, thanks for participating. This study is not a test of intelligence or personality. Your task is just to win points by pressing the button. Whenever you win one point you will hear a sound, whenever you don't win you won't hear anything; you will never lose points, however. The more sounds you hear, the more points you win. Press the button to start the experiment!

The confidence threshold for data recording during the session was set to a value of .80, as explained above.

Data analysis.

Before analyzing the gaze data reported by Pupil, we excluded off-screen data points and performed cloud-center correction by blocks of 1,000 frames, as explained in **Optimizing Data Quality**. A subject was considered to be looking at a stimulus (be it the square on the left or the circle on the right) whenever the distance between the subject's gaze location and the center of this stimulus was less than half the horizontal distance between the centers of the two stimuli.

Because the number of schedule components varied slightly across phases, when analyzing each phase we discarded the data from any component past Component 36 (that is, past the first nine full four-component blocks). Examining the raw rates of gazing at each stimulus in these additional components showed them to be merely the continuation of what happened in the preceding block. Thus, excluding these few data points simplified our analysis without impacting the overall pattern of results.

Finally, two students (P5 and P8) proved insensitive to the changes of reinforcement schedules across phases, showing no evidence at all of button-pressing differentiation in Phase B (a differentiation which we took as a prerequisite for any possible effect of reinforcement on observing). These subjects showed no evidence of differential looking either. Accordingly, their results were discarded, and data analysis proceeded only with respect to the remaining six subjects. Main data analyzes were done in R (R Core Team, 2017).

Results and Discussion

Figure 4 shows the proportion of button presses in S+ (that is, the number of button presses in S+ divided by the total of button presses in S+ and S-) along successive blocks of four components during the experiment. Black dots represent group means, whereas dotted vertical lines indicate ranges. Starting below .50, the mean proportion of button presses in S+ reached indifference by the end of the first baseline phase (A), while button pressing was reinforced in the presence of all stimuli. Mean proportions quickly approached 1 in Phase B, when button pressing was reinforced in the presence of S+ and extinguished in the presence of S-, to return to indifference in the final phase (A), when button-pressing was again reinforced non-differentially. Notable differences of vertical range along successive blocks reflect the fact that not all subjects changed their behavior simultaneously across phases. Thus, at the start of Phase B one subject (P3) lagged behind the others, changing his behavior only by the fourth block in the phase, whereas in the final phase another subject (P6) persisted in not responding in S- during most of the four-component blocks.

The panels of Figure 5 demonstrate that our subjects differed in ocular behavior even during their first baseline phase. Each panel shows a subject's left-gaze proportion (that is, the number of gazes toward the left stimulus divided by the number of gazes toward the left and right stimuli) along successive components of the initial A phase, plotted separately for the components that followed a stimulus change on the left (odd-numbered components, solid curves) and for the components that followed a stimulus change on the right (even-numbered components, dotted curves). One would expect a subject who tracked stimulus changes in baseline to have a higher proportion of left gazes after a stimulus change on the left than after a stimulus change on the right. Accordingly, the panels in Figure 5 have been ordered in terms of the mean difference between a subject's solid and dotted curves, a difference that served as our index of stimulus tracking. This index averaged .28 and ranged from .84 (subject P1, top panel) to -.21 (subject P7, bottom panel). Notice that even though our subjects differed widely in terms of stimulus tracking, all subjects started the baseline with a solid curve above the dotted curve (Figure 5). Thus, the main difference between the subjects with a high and a low tracking index was that the former (e.g., P1, P4) had their solid and dotted curves well separated until the end of the baseline, whereas the latter (e.g., P6, P7) showed solid- and dotted-curve crossovers before the end of the baseline. If we assume that at least some of the behavior in Figure 5 reflects elicitation by stimulus change and that elicited gazing is susceptible to habituation (e.g., Colombo & Mitchell, 2009), then the panels of Figure 5 suggest that different subjects had different habituation rates.

The panels of Figure 6 demonstrate that our subjects also differed in the extent of their gaze switching during extinction. In the making of Figure 6, we counted a gaze switch whenever a subject looked at different stimuli in two successive frames. Each panel shows a subject's switching proportion (that is, the number of gaze switches during S-, divided by the number of gaze switches during S- and during S+) along successive four-component blocks in each of the three phases of the experiment. The switching proportions in each phase were smoothed via a 3-point moving average to emphasize main trends. One would expect a subject sensitive to stimuli in relation to the extinction of button-pressing to show an increased switching proportion in Phase B; accordingly, the panels in Figure 6 have been ordered in terms of each subject's mean switching proportion in this phase, a quantity that served as our index of extinction-related switching. This index averaged .59 and ranged from .78 (subject P1, top panel) down to .43 (subject P6, bottom panel). Extinction has long been known to increase the variability of operant behavior (e.g. Antonitis, 1951), but Figure 6 documents the effect of the extinction of one response (button pressing) on the variability of another response (gazing). This effect of extinction was evident at the within-subject level only in P1, P2, and P4, however.

Figure 7 shows that our subjects' degree of extinction-induced switching in Phase B (Figure 6) could have been predicted from their stimulus-tracking behavior in the initial baseline (Figure 5). In this figure, the index of baseline stimulus tracking (that is, the mean difference between the solid and dotted curves in Figure 5) appears on the *x* axis, whereas the mean extinction-induced switching proportion in Phase B (as defined in Figure 6) appears on the *y* axis, each dot representing a different subject. The two measures were highly correlated at the group level ($r(4) = .91$, $p = .01$), with a slope of ordinary least-square regression equal to 0.33 (Figure 7).

Finally, Figure 8 shows, for each subject, the proportion of gazes on the left (that is, the number of gazes on the left divided by the number of gazes on the left or on the right) during S+ along the successive four-component blocks of Phases A, B, and A. As in Figure 6, proportions were smoothed with a 3-point moving average. Theories of conditioned reinforcement predict that this proportion should have increased in Phase B. Accordingly, we defined our index of S+ observing as the difference between the mean left-gaze proportion during S+ in Phase B and the mean left-gaze proportion during S+ in both A Phases. The individual panels in Figure 8 have been ordered in terms of this index, from higher (Subject P4, top panel) to lower (Subject P6, bottom panel). The index was positive in Subjects P4, P1, P3, P7, and P2 (mean = .09, range = [.05, .18]) and negative in the case of Subject P6, with a value of -.11. For the five subjects who showed a positive index of left-stimulus observing during S+, computing the same index during S- returned values that were lower than the former in all cases (mean = .02, range = [-.06, .11]; paired *t*-test, $t(4) = 2.76$, $p = .05$).

These results provide some evidence for conditioned-reinforcement effects that were specific to the observing of S+ in Phase B. The evidence at the group level is weak, however, which is unsurprising given our use of continuous reinforcement for button pressing. As explained in our Introduction, this was done to eliminate superstitious responding, but entailed that the points delivered after each response were able to overshadow or block visual stimuli (cf. Mackintosh, 1977). Another factor that may have contributed to the weakness of the effects was the fact that visual changes on the right always competed for stimulus control with visual changes on the left. Be that as it may, our subjects' observing of S+ in Phase B correlated positively with their stimulus tracking in baseline ($r = .57$). Although not significant ($r(4) = .57$, *p* = .24), this correlation is consistent with the results of Figure 7, which shows a strong (*r* = .91) relation between stimulus tracking in baseline and extinction-induced gaze switching in Phase B. Studies on the conditioned reinforcement of the human gaze often exhibit marked between-subject variability (e.g., Tomanari et al., 2007, Figure 5, p. 36); our findings suggest that at least part of this variability may be traced to initial differences in the subjects' stimulus-tracking repertoires. Studying the latter will not make variability vanish, but might make it easier to understand.

In any case, our study was not designed to gather the most general or robust findings about observing, but to establish proof-of-concept for using the Pupil platform in behavior analysis. The eye-tracking system can be seen to generate consistent results (e.g., Figure 7) provided appropriate coordinate corrections are employed. Aside from its low cost, the most attractive feature of Pupil consists in the possibility of data-analysis automation along the lines of what we have been describing. Two important limitations of the present report should be noted, however. First, we tested Pupil on a GNU/Linux operating system. We did not test the platform on Windows or OSX, and therefore we cannot comment on how Pupil would perform on these systems. Second, our setup involved a chin rest to maintain the subjects' head stationary. The spatial errors apparent in gaze estimation could be corrected, but pending further improvement in the platform, their existence suggests caution with using Pupil in eye-movement research on freely moving people. Within these limits, we hope that the present report will encourage a growing number of researchers to study the human gaze in relation to stimulus control and reinforcement.

Footnote 1. We advise against buying a regular headset from an optician's store, as Pupil Labs' headsets have been optimally designed with mountpoints ready at the locations needed by the cameras. Besides, academic researchers can obtain genuine Pupil-Labs monocular headsets at a reduced price from the 3-D printing store, ShapeWays (https://www.shapeways.com.)

Footnote 2. A camera that follows the USB-video-class (UVC) specification will allow video streaming via the operating system's standard USB driver.

References

Antonitis, J. J. (1951). Response variability in the white rat during conditioning, extinction, and reconditioning. *Journal of Experimental Psychology*, *42*, 273-281. http://dx.doi.org/10.1037/h0060407

Colombo, J., & Mitchell, D. W. (2009). Infant visual habituation. *Neurobiology of Learning and Memory*, *92*, 225–234. https://doi.org/10.1016/j.nlm.2008.06.002

Dinsmoor, J. A. (1983). Observing and conditioned reinforcement. *Behavioral and Brain Sciences*, *6*, 693-728. (Includes commentary) https://doi.org/10.1017/S0140525X00017969

Dinsmoor, J. A. (1985). The role of observing and attention in establishing stimulus control. *Journal of the Experimental Analysis of Behavior*, *43*, 365-381. https://doi.org/10.1901/jeab.1985.43-365

Dube, W. V., Dickson, C. A., Balsamo, L. M., O'Donnell, K. L., Tomanari, G. Y., Farren, K. M., ... McIlvane, W. J. (2010). Observing behavior and atypically restricted stimulus control. *Journal of the Experimental Analysis of Behavior*, *94*, 297–313. https://doi.org/10.1901/jeab.1985.43-365

Dube, W. V., Lombard, K. M., Farren, K. M., Flusser, D. S., Balsamo, L. M., & Fowler, T. R. (1999). Eye tracking assessment of stimulus overselectivity in individuals with mental retardation. *Experimental Analysis of Human Behavior Bulletin*, *17*, 267–271.

Ester, M., Kriegel, H.-P., Sander, J., & Xu, X. (1996). A density-based algorithm for discovering clusters in large spatial databases with noise. In *Proceedings of the Second International*

Conference on Knowledge Discovery and Data Mining (pp. 226–231). Menlo Park, CA: AAAI Press.

Fantino, E. (1977). Conditioned reinforcement: Choice and information. In W. K. Honig & J. E. R. Staddon (Eds.), *Handbook of operant behavior* (pp. 313-339). Englewood Cliffs, NJ: Prentice-Hall.

Holland, J. G. (1958). Human vigilance. *Science*, *128*, 61-67. https://doi.org/10.1126/science.128.3315.61

Huziwara, E. M., de Souza, D., & Tomanari, G. Y. (2016). Patterns of eye movement in matching-to-sample tasks. *Psicologia: Reflexão E Crítica*, *29*, 2. https://doi.org/10.1186/s41155-016-0010-3

Kassner, M., Patera, W., & Bulling, A. (2014). Pupil: an open source platform for pervasive eye tracking and mobile gaze-based interaction. In *Proceedings of the 2014 ACM International Joint Conference on Pervasive and Ubiquitous Computing: Adjunct Publication* (pp. 1151–1160). Seattle, WA: ACM Press.

Mackintosh, N. J. (1977). Stimulus control: Attentional factors. In W. K. Honig & J. E. R. Staddon (Eds.), *Handbook of operant behavior* (pp. 481-513). Englewood Cliffs, NJ: Prentice-Hall.

Ono, K. (1987). Superstitious behavior in humans. *Journal of the Experimental Analysis of Behavior*, *47*, 261-271. https://doi.org/10.1901/jeab.1987.47-261

Paeye, C., & Madelain, L. (2011). Reinforcing saccadic amplitude variability. *Journal of the Experimental Analysis of Behavior*, *95*, 149–162. https://doi:10.1901/jeab.2011.95-149

Palmer, D. (2010). Behavior under the microscope: Increasing the resolution of our experimental procedures. *The Behavior Analyst*, *33*, 37–45. https://doi.org/10.1007/BF03392202

Perez, W. F., Endemann, P., Pessôa, C. V. B. B., & Tomanari, G. Y. (2015). Assessing stimulus control in a discrimination task with compound stimuli: Evaluating testing procedures and tracking eye fixations. *Psychological Record*, *65*, 83–88. https://doi.org/10.1007/s40732-014-0092-1

Picanço, C. R. F. (2017). *Stimulus Control* [Free Pascal computer program]. https://doi.org/10.5281/zenodo.591263

R Core Team (2017). *R: A language and environment for statistical computing.* R Foundation for Statistical Computing, Vienna, Austria. https://www.R-project.org/.

Schroeder, S. R., & Holland, J. G. (1968). Operant control of eye movements during human vigilance. *Science*, *161*, 292-293. https://doi.org/10.1126/science.161.3838.292

Schroeder, S. R., & Holland, J. G. (1969). Reinforcement of eye movement with concurrent schedules. *Journal of the Experimental Analysis of Behavior*, *12*, 897-903. https://doi.org/10.1901/jeab.1969.12-897

Shahan, T. A., & Cunningham, P. C. (2015). Conditioned reinforcement and information theory reconsidered. *Journal of the Experimental Analysis of Behavior*, *103*, 405-418. https://doi.org/10.1002/jeab.142

Steingrimsdottir, H. S., & Arntzen, E. (2016). Eye movements during conditional discrimination training. *Psychological Record*, *66*, 201-212. https://doi.org/10.1007/s40732-015-0156-x

Tomanari, G. Y. (2004). Human observing behavior maintained by S+ and S-: Preliminary data. *International Journal of Psychology and Psychological Therapy*, *4*, 155-163.

Tomanari, G. Y., Balsamo, L. M., Fowler, T. R., Lombard, K. M., Farren, K. M., & Dube, W. F. (2007). Manual and ocular observing behavior in human subjects. *European Journal of Behavior Analysis*, *8*, 29-40. https://doi.org/10.1080/15021149.2007.11434271

Wagner, G.A., & Morris, E.K. (1987). "Superstitious" behavior in children. *Psychological Record*, *37*, 417-488. https://doi.org/10.1007/BF03394994

Figure Captions

Figure 1. A view of the gaze-recording setup used in our laboratory. The hardware shown in the figure is identical to that employed in our study, with the exception of a more recent eye-camera model. The figure shows the Pupil headset with the eye camera, the world camera, the USB cables clipped to the subject's clothing, and the chin rest. The projector we used to display experimental stimuli is visible in the background.

Figure 2. An example of gaze estimation by Pupil. The data are from the baseline session of one of the subjects in our study. The top panel shows the screen in normalized Cartesian coordinates, with the contours of the stimuli employed (a square and a circle) and cross-hairs at the center of the screen. Each dot in the cloud represents a single gaze location as reported by Pupil. The inset in the panel shows how the center of the cloud drifted during the session. Bottom panel: The same data after cloud-center correction. See text for more detailed explanations.

Figure 3. A schematic of hardware and software in a typical gaze-recording study. Among the software components (rectangles with rounded corners) needed to run a session, those that are actually part of the Pupil platform are identified in bold. These are Pupil Capture and Pupil Player. The remaining software is either part of the operating system (in the case of system time) or is custom software. Our coordinate-transformation plugin was written in Python, but the programs needed to present stimuli and to analyze data can be written in any language of convenience.

Figure 4. Button-pressing proportion (computed as the number of button presses in S+ divided by the total of button presses in S+ and S-) as a function of successive blocks of four components. Black dots indicate group means. Vertical dotted lines indicate the range of the data.

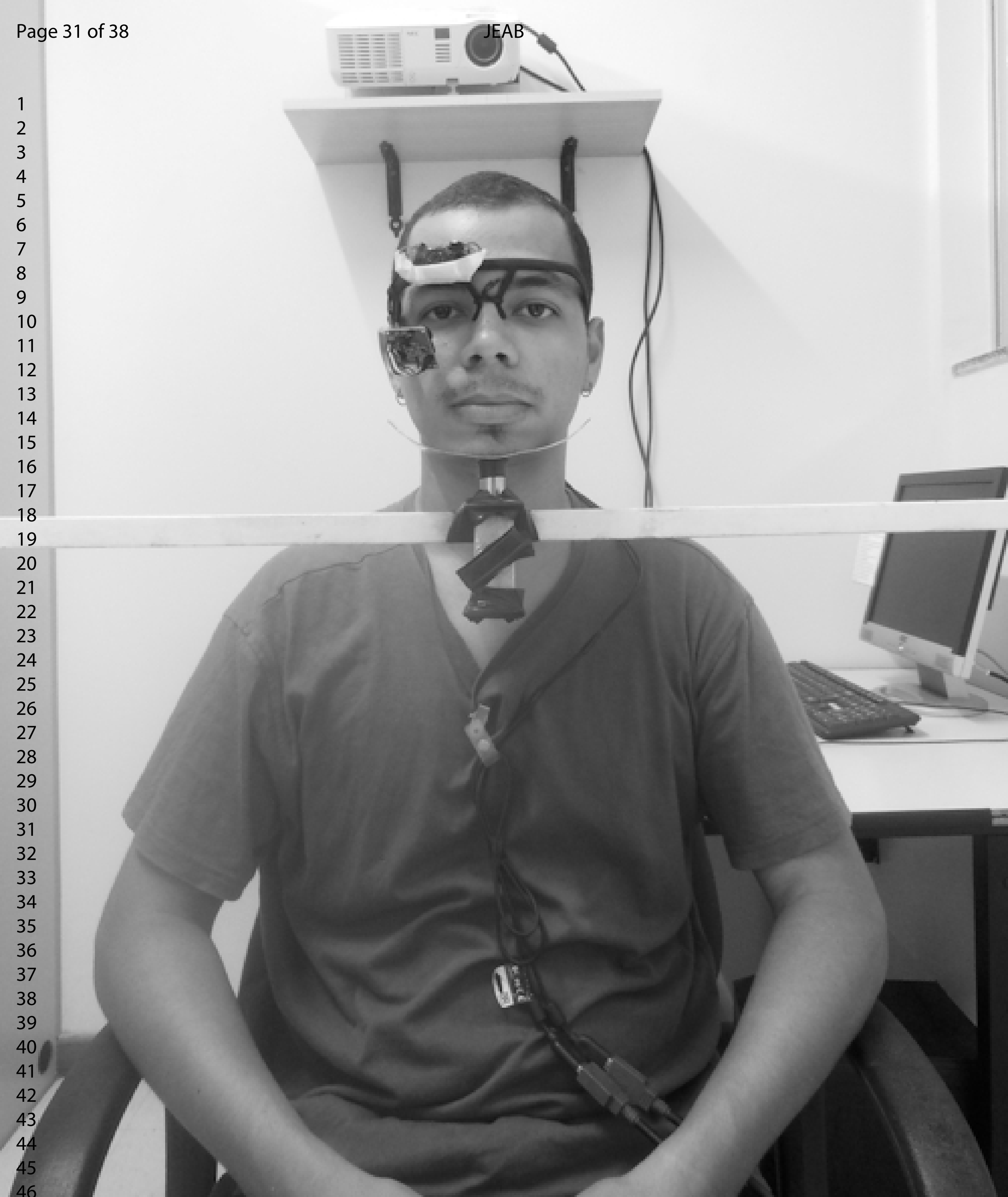
Figure 5. Proportion of left-stimulus gazes along successive components in the initial baseline (first A Phase). Each panel shows the data from a different subject. Proportions are plotted separately for the components that followed a stimulus change on the left (odd-numbered components, solid lines) and for the components that followed a stimulus change on the right (even-numbered components, dotted lines). The panels are ordered in terms of the mean difference between the solid and dotted curves, from higher (subject P1) to lower (subject P7). As explained in the text, this difference served as our measure of stimulus tracking in baseline.

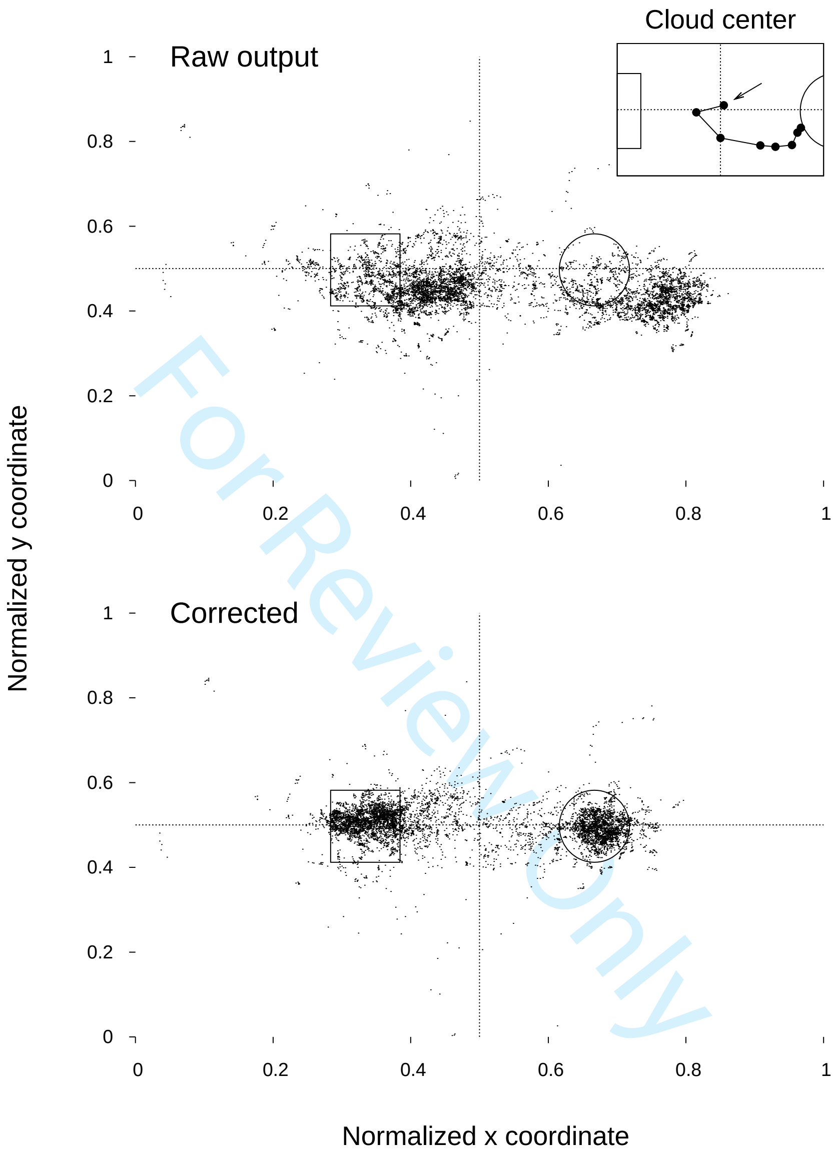
Figure 6. Switching proportion along successive four-component blocks in the session. This proportion was defined on each block as the number of gaze switches during the two components with a blue square on the left, divided by the total number of gaze switches during the four components. We expected this proportion to increase in Phase B, due to the extinction of button pressing in the presence of the blue square. The panels are ordered according to the average switching proportion in Phase B, from higher (subject P1) to lower (subject P6). The data were smoothed with a 3-point moving average to emphasize main trends.

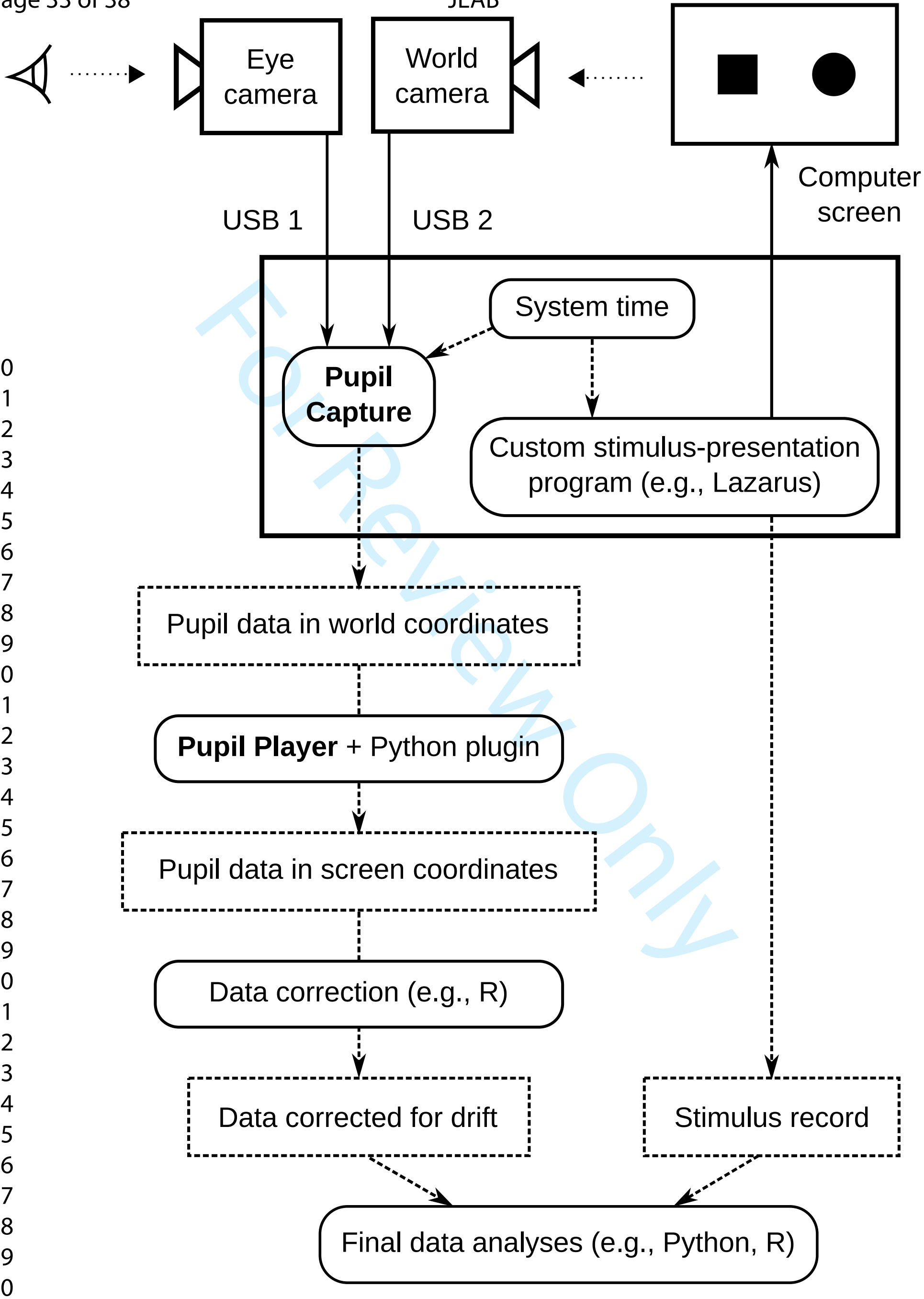
Figure 7. The relation between our measures of baseline stimulus tracking and extinction-induced switching in Phase B. Each dot represent a different subject. The solid line was fitted by the method of ordinary least squares.

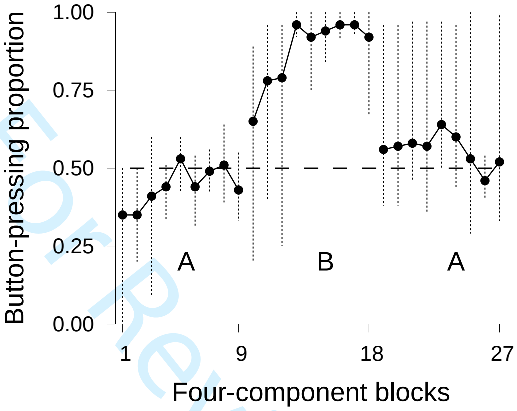
Figure 8. Proportion of left-stimulus gazes in the presence of the red square (S+) along successive four-component blocks in the session. We expected this proportion to increase in

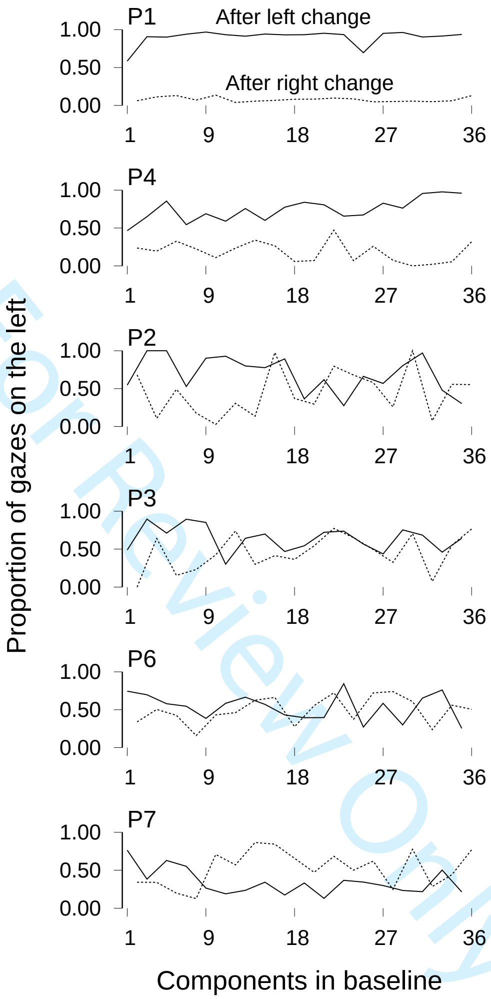
Phase B, due to the extinction of button pressing in the presence of the blue square (S-). The panels are ordered according to the difference between the mean proportion in Phase B and the mean proportion in the preceding and succeeding A Phases (see text for more explanation). As in Figure 6, the data were smoothed with a 3-point moving average. Each panel shows the results of a different subject.

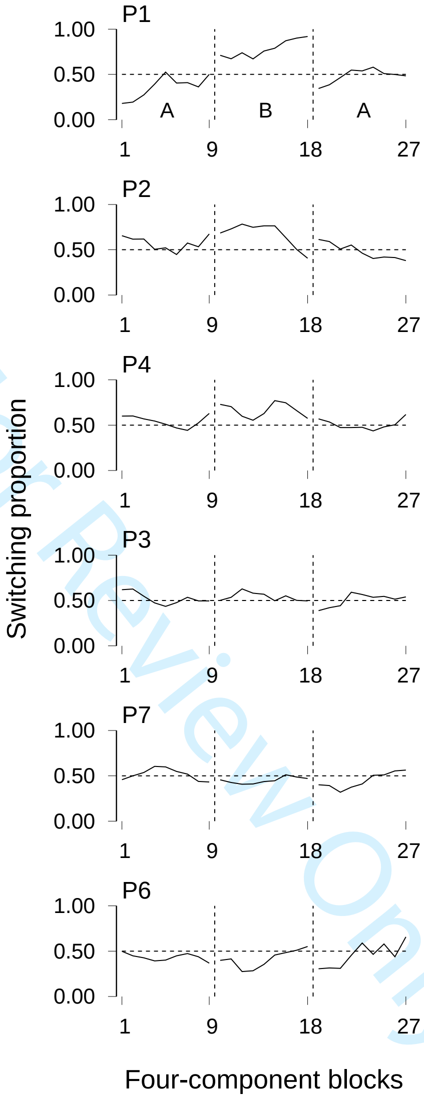


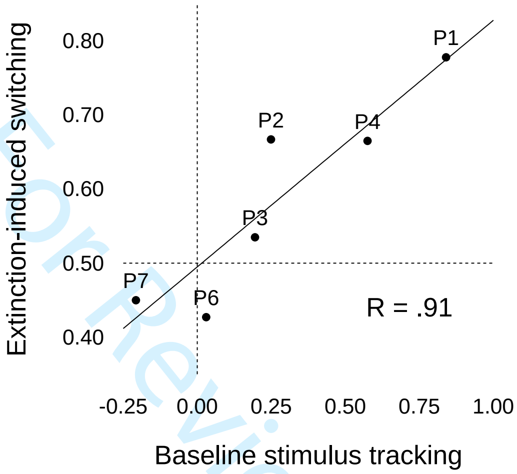

Cloud center
Raw output
Corrected
Normalized y coordinate
Normalized x coordinate
0
0.2
0.4
0.6
0.8
1

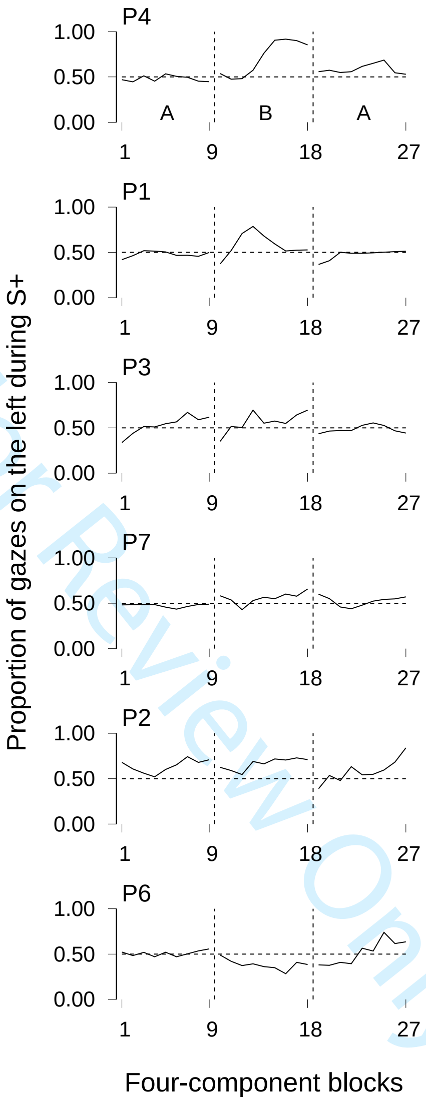

Eye camera
World camera
USB 1
USB 2
Computer screen
System time
Pupil Capture
Custom stimulus-presentation program (e.g., Lazarus)
Pupil data in world coordinates
Pupil Player + Python plugin
Pupil data in screen coordinates
Data correction (e.g., R)
Data corrected for drift
Stimulus record
Final data analyses (e.g., Python, R)


Button-pressing proportion
1.00
0.75
0.50
0.25
0.00
A
B
A
1
9
18
27
Four-component blocks


P1
After left change
After right change
1.00
0.50
0.00
1
9
18
27
36
P4
P2
P3
P6
P7
Proportion of gazes on the left
Components in baseline


P1
1.00
0.50
0.00
A
B
A
1
9
18
27
P2
P4
P3
P7
P6
Switching proportion
Four-component blocks


Extinction-induced switching
0.80
0.70
0.60
0.50
0.40
P1
P2
P4
P3
P7
P6
R = .91
-0.25
0.00
0.25
0.50
0.75
1.00
Baseline stimulus tracking


P4
1.00
0.50
0.00
A
B
A
1
9
18
27
P1
P3
P7
P2
P6
Proportion of gazes on the left during S+
Four-component blocks