Improving the welfare of laboratory macaque monkeys and pre-training them for cognitive visual tasks

By implementing variable reward schedules in touchscreen tasks

Diogo Alexandre Rocha Moreira

Bachelor Thesis for Biomedical Sciences 2022 – 2023

Laboratory of Prof. Michael Schmid
Under supervision of Jaime Daniel Cadena Valencia and Marcus Haag

Faculty of Science and Medicine
University of Fribourg, Switzerland

Abstract

Laboratory animals are widely used in pharmaceutical and biomedical research. Due to their structural and functional similarity of their visual cortex, macaque monkeys are often used to investigate questions in visual neuroscience. To address research questions in this domain, monkeys undergo extensive training protocols to familiarize them with laboratory environments and complex cognitive tasks. However, current training methodologies are time-consuming, prone to unexpected delays. This study, aimed to simultaneously improve animal welfare and training efficiency in pre-translational visual neuroscience research involving rhesus monkeys. To do so, we designed a touchscreen task for the XBI (a specific tablet designed for experiments with monkeys) and associated it with different positive reinforcement schedules in order to propose an unsupervised task that can be implemented in the monkeys' home enclosure to increase environment enrichment and pretrain monkeys for future experimental paradigms. For pre-training to be successful, the animals need to be highly engaged with the task. For this purpose, we aim to study the effect of variable reward schedules compared to fixed reward schedules on engagement. To standardize the pre-training for different monkeys, we designed the outline of a protocol that adapts the difficulty of the task depending on the monkeys' performance and runs them through different reward schedules. Preliminary testing on a trained monkey demonstrated the feasibility of the touchscreen task; however, adjustments to the difficulty curve were identified as necessary. To optimize the task and protocol further, as well as determining the most engaging reward schedule, comprehensive observations of monkey-task interactions during multiple experiment sessions are still required.

Introduction

Visual neuroscience is an interdisciplinary field that aims to understand the intricate workings of the visual system, including its cognitive aspects. The study of vision and cognition requires complex and demanding experimental tasks that aren't viable for all kinds of animal models.

Moreover, in our laboratory, we carry out pre-translational studies with the aim of developing therapies for blindness, which accentuates the need for an animal model similar to the human in terms of the visual system, because the greater the difference between the model studied and the human, the more complicated it is to translate the research results.

For these reasons, non-human primates are a suitable model for our research given that they are capable of performing complex visual tasks and have many similarities with humans in terms of cognition and visual system. Some of the similarities being : A highly developed associative cortex (Hofman 2014), a similar contrast discrimination to humans (Kiorpes 2016) and forward facing eyes.

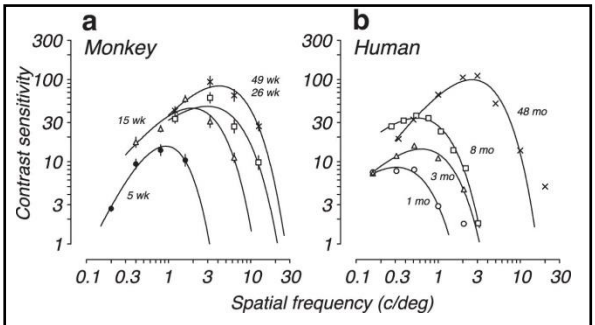

Figure 1 : (Kiorpes 2016) Contrast sensitivity development in macaque monkeys and humans. This figure shows the similarities between monkeys and humans in contrast sensitivity across multiple developmental stages. Although not identical, the contrast discrimination capacity is very close which supports the employment of monkeys in visual neuroscience research.

Challenges in Monkey-Based Research

There are many challenges with the use of NHPs in neuroscience, among them is the training protocol they are submitted to in order to be prepared for the research experiments.

In the next section we will detail the different steps of monkey training in view of visual neuroscience experiments, we will also discuss its current major issue and propose an initial solution to increase training efficiency.

Lab training

The training protocol of NHPs in our laboratory spans 4 major steps (represented in the diagram below) :

1) When the monkeys get to the lab, as per Switzerland regulations, they are kept under *quarantine* for a duration of two weeks to a month to ensure safety of the lab environment.

2) After the quarantine comes a *habituation* period that lasts approximately one month, this step focuses on the acclimatization of the macaque monkeys to the lab facilities (their home enclosure and the caretakers).

3) The third step consists of multiple types of training necessary for the monkey to work in the lab *environment*, this step also lasts one month. First the animals are trained to get from the big enclosure to a smaller and easier to handle cage. Afterwards, they are taught to enter the "primate chair" (a special chair used to mobilize NHPs to the task setup) (McMillan, Bloomsmith, and Prescott 2017), stick their head out of the chair and finally start familiarizing with the experiment room.

4) The final step ranges from 1 to several months, where the NHPs are trained for *specific tasks* of the lab (i.e. contrast discrimination tasks or visual search tasks). The length of this step will vary greatly depending on the tasks and different surgeries scheduled.

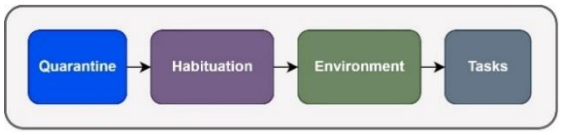

Figure 2 : Diagram representing the 4 different steps of monkey laboratory preparation in chronological order.

As we have seen, laboratory monkey's preparation is an arduous and lengthy process involving several stages, the order of which must be respected to ensure correct training progress. Furthermore, the training protocol may be subject to unforeseen extensions depending on the performance of the subjects being trained (i.e. if the monkey has a hard time understanding the task, it will take a longer time to conclude its training).

A solution would be to implement pre-training with an easy task that the monkeys can perform inside their home environment in an unsupervised fashion, meaning that they could interact with it at their own pace. The inclusion of this autonomous task would not only engage the animals in training basic interactions early on, but also participate in increasing the environmental enrichment throughout the training timeline.

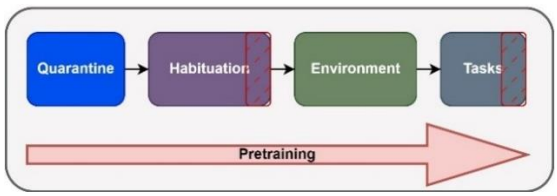

Figure 3 : Diagram representing the beneficial effects of pre-training as a parallel procedure to the original monkey preparation steps. The red hashes design the decrease in time for the habituation and specific tasks steps.

Positive reinforcement

Animal training is commonly done using positive reinforcement where we reinforce a target behavior of the animal by providing rewards in consequence to it which increases the probability of such behavior being repeated in the future ("APA Dictionary of Psychology" n.d.).

Positive reinforcement is one type of reinforcement described in operant conditioning, which was coined by Skinner in 1937 and based upon Edward Thorndike's "Law of Effect" (1905). Skinner proposed this theory believing that it was not necessary to look at the internal workings and motivations in order to explain behavior, alternatively he focused on the consequences of such behavior (Staddon and Cerutti 2003). These consequences are termed as positive reinforcers if they reward the behavior or negative reinforcers if they punish it.

Schedules of reinforcement

In the book "Schedules of reinforcement" from Skinner and Ferster (1957), different types of positive reinforcement schedules are explained. They can be used to reinforce animal behaviors for cognitive experiments with variable effects on engagement.

(A) *Continuous reinforcement* (Skinner and Ferster 2015) is the usual schedule associated with positive reinforcement. Every time that the animal answers correctly, he gets a reward (i.e. a food pellet). This was the main schedule used for animal training until Skinner found it to be wasteful and expensive to reward every trial across hundreds or thousands of trials since it amounted to a vast quantity of food pellets that needed to be replenished each day. Thus 2 new types of schedules were discovered :

(B) *Ratio reinforcement* (Skinner and Ferster 2015) where the animal only gets a reward after providing a predefined amount of target behaviors, so to speak, a specific ratio. This type of reward schedule is accompanied by a high response rate from the subject since the more correct trials, the closer he gets to another reward.

(C) *Interval reinforcement* (Skinner and Ferster 2015) is a schedule where the

rewards are limited by a time delay. The animal only gets a reward if a certain predefined interval has passed since the last reward. This schedule tends to be associated with a moderate response rate as a higher rate of answers doesn't increase the likelihood of reward.

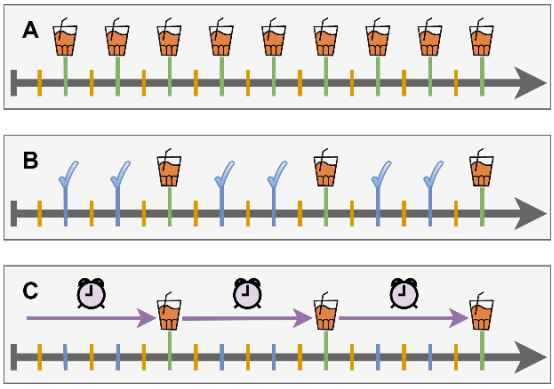

Figure 4 : Illustration of the 3 main reinforcement schedules based upon their description on **(Skinner and Ferster 2015)**. (A) Continuous reinforcement. (B) Ratio reinforcement. (C) Interval reinforcement. The green bars being correct trials that were rewarded, the blue bars are correct trials not rewarded and the orange bars being the start of each trial.

The schedules can be further separated by the randomization of their parameters :

Fixed schedules (Skinner and Ferster 2015) have their main parameter (ratio/interval) stay constant after each reward. This makes it likelier for the subject to understand the type of schedule being used and react accordingly. Once the schedule is understood, it is likely for the animal to make pauses after getting a reward. The effect is even higher for interval schedules as the subject only needs to wait for the interval to finish before providing another response.

Variable schedules (Skinner and Ferster 2015) have their main parameter randomized after each reward. Thus, making it harder for the subject to understand the exact type of schedule being used during training. Decreasing the predictability of the schedule may induce a steady response rate from the subjects. We could explain this possibility with a simple example : When taking the bus, we tend to buy a ticket because we cannot be certain of the presence of a controller or not (steady response rate). If we could predict its presence in advance, then we might be inclined to only buy the ticket when the controller will appear (less steady response rate).

The advantage of variable schedules over fixed schedules on engagement was observed by Skinner during his experiments (the response rate was indeed higher when using variable reward schedules compared to fixed ones) and is also a principle frequently used in casinos, by adding variability to the rewards, customers are more engaged to play and come back to the facilities.

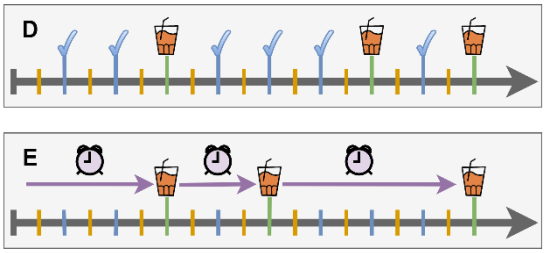

Figure 5 : Illustration of the variable versions of reinforcement schedules. (D) Variable ratio reinforcement. (E) Variable interval reinforcement.

Hypothesis and aims of the study

The main issues that were addressed concerning the employment of monkeys are : 1) Lengthy training periods. 2) Environmental enrichment. 3) Movement restriction. 4) Fixed reward schedules' engagement.

Previous work

In a study published in Journal of Neurophysiology (Berger et al. 2018), Berger et al. aimed to develop a standardized and automated training paradigm for Non-Human Primates in cognitive neuroscience research. The authors aimed to develop a training system

that would help NHPs learn complex cognitive tasks in an unsupervised, self-paced manner. The development of such a system could reduce the time invested by the caretaker training the animal while increasing the animals' motivation and welfare all while standardizing training protocols.

The study showed that unsupervised training can be an alternative to train monkeys in complex tasks used for cognitive neuroscience research, however the progress tends to be slower.

In our study we designed an unsupervised task without the automated difficulty adjustment not to create an alternative to the usual training protocols but to supplement them and optimize the training timeline.

Aims of the study

For visual neuroscience projects with monkeys to be more efficient we want to work on the aforementioned issues.

Thus this study wants to improve upon two main principles : Animal welfare and Training efficiency.

Animal welfare is always a major concern for all research groups using animal models, however, many factors during research will tend to decrease said welfare such as a lack of environmental enrichment, the movement restriction needed for some experiments and even long training periods themselves. With this study, we want to address the challenge of animal welfare by providing an unsupervised task that will increase environmental enrichment, delay the need for movement restriction training and decrease the length of training protocols.

Training efficiency is essential in a case where we have to train animals for long periods of time. As we've mentioned previously, the current training protocol for rhesus monkeys' training is very time consuming and it risks unexpected extensions depending on the monkeys' understanding that can increase the overall experiment time and thus make it harder to ensure the projects' conclusion. By providing an unsupervised pre-training protocol in the home environment of the animals, the training phase and overall experiment time might decrease, thus increasing the maneuverability of the study timeline and so being able to adapt to unexpected situations without risking to run out of time and failing the project.

Hypotheses

To accomplish our aforementioned aims we have 3 hypotheses :

Variable reward schedules produce higher engagement than fixed schedules.

- Null hypothesis : There is no significant difference in engagement between variable and fixed reward schedules.
- Analysis : Compare the average engagement levels between a group of monkeys trained with variable schedules and a group trained with fixed schedules by using a T-test.

Pre-training increases efficiency of the training timeline.

- Null hypothesis : Pre-training doesn't decrease the training and overall experiment time.
- Analysis : Compare the time needed to train monkeys with pre-training with the time needed for previous training timelines.

An unsupervised home cage-based task increases welfare by providing better environmental enrichment.

- Null hypothesis : The autonomous task doesn't increase welfare.

- Analysis : Track and compare the behavior of the monkeys during the training timeline between a group exposed to the unsupervised task and a group exposed only to normal training.

Methods

Experimental approach

To test our hypotheses, a touchscreen task specifically designed for the XBI was designed by us. Within the task, various types of reward schedules, which can be easily modified between sessions, were implemented. The task, associated with the XBI, will be connected to an individual cage, allowing the observation of the effects of the current difficulty curve, the need for additional protocol optimizations, and the testing of different schedules in regards to our hypotheses.

As this project is mainly based upon the development of the touchscreen task and the tools needed to analyze its data, the test of aforementioned hypotheses is not concluded, however, the structure needed for a future study to do so is established.

Subject selection

For this study to be carried out, rhesus monkeys were selected as the main model. Stan, one of the two trained monkeys from our lab was employed as the tester for the XBI touchscreen task, the decision being taken depending on the availability of said subject.

Please note that although 5 additional naïve monkeys were later introduced to the lab, they were not included in this study as it was not feasible to work with them at that time.

Materials

The eXperimental Behavioral Instrument (XBI) (Calapai et al. 2017), a specific touchscreen used for cognitive neuroscience experiments on monkeys, served as the main tool of our project.

Two main interfaces, the AI (Animal interface) and the CI (Control interface), are contained within the XBI, and they can either be associated inside a main frame or separated.

The *AI*, the part with which the animal interacts, features a funnel shaped middle section that tapers down to the size of the actual touchscreen. A metal tube, with a small opening through which the juice reward is dispensed, is found across the funnel, and the tube can be rotated for the monkey to adjust it as he pleases. The reward is controlled by peristaltic pumps that are connected to fluid recipients placed on the sides of the AI metal frame. The whole AI is encompassed by a metal frame, making it sturdy and durable for long-term work. Additionally, there is a camera to record the animal, and speakers are used to deliver sounds during the experiment.

The CI, the part that is interacted with by the researchers, contains all the hardware and software required to operate the AI. In our case, it is connected to the main frame and is situated behind the AI. The touchscreen is thus connected to a computer that controls the Local Area Network and the flow of the task through a software called MWorks. Thanks to this disposition, the task can be controlled by the researcher while it is being performed by the animal, although in our project, the task will be automated to minimize the need for supervision.

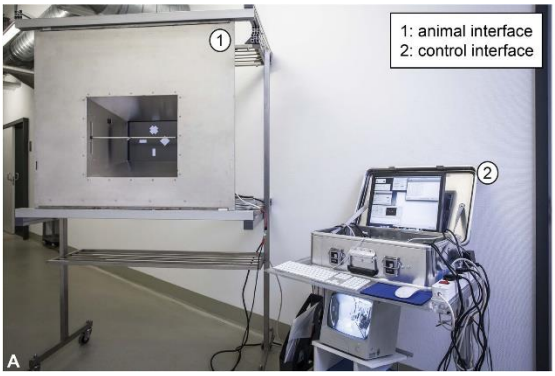

Figure 6 : (Calapai et al. 2017) Image of the XBI multipart system. (1) Animal interface. (2) Control interface. Here we see the separated configuration but the AI and CI can also be merged into one structure.

MWorks

An open source platform called MWorks is used for designing and running behavioral experiments in the domains of psychology and neuropsychology. Tools are provided by MWorks to control and follow the flow of the experiment during the sessions, and a broad range of input and output devices (i.e. monitors, eye trackers, speakers) are supported ("The MWorks Project" n.d.).

There are two softwares that make up MWorks : MWorks client, where the task is chosen and modified, and MWorks server, which receives the output from the client and manages the experiment workflow (stimuli presentation, data acquisition), and then sends the collected data to the client.

MWEL

The experiments run on this platform are based on MWEL (MWorks Experiment Language), a programming language specifically designed for MWorks. A script containing the MWEL code that defines the parameters, macros, and protocol of the experiment is first written to create a MWorks experiment. The previous method of creating XML-based experiments, which was more interface heavy and did not require programming experience, is replaced by this alternative. By using MWEL, complex experiments with a higher degree of flexibility and customization can be easily developed by the researcher ("Experiment Language (MWEL) — MWorks 0.13.Dev Documentation" n.d.)

Early design of the task

The project revolves around the design of a touchscreen task for pre-training purposes. However, the actual design of the task underwent significant changes from its early stages.

The first idea of the design was inspired by the visual search task developed by Professor Gijsbert Stoet. It involves the presentation of multiple distractors with a specific shape and color on the screen, along with one target with a different shape and color that needs to be touched by the subject. The positions of the stimuli would be randomized between each trial.

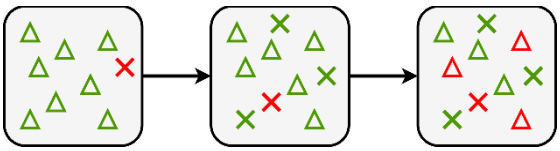

Figure 7 : Illustration of the different versions of the early design of the task.

The task would then be associated with a staircase algorithm that would adapt the difficulty depending on the performance of the monkey. If 80% or more of all trials were correct across 50 trials, then one level of difficulty would be increased. If 20% or less were correct, then one level of difficulty would be decreased. The levels of difficulty were represented as different versions of the task as depicted in figure 7.

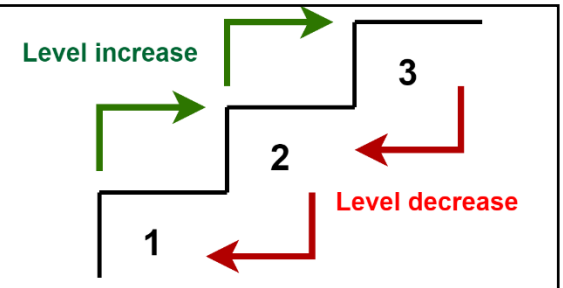

Figure 8 : Illustration of the flow of a staircase algorithm.

Although this design is very attractive for studying attention, it was not suitable for our project as it was very complex to code and implemented too many variables that would make it hard to isolate the effects of reward schedules on engagement later on. Another reason for scrapping this idea is that it did not relate to the tasks being performed in our lab, so as a result, we would not observe as many benefits in these tasks with pre-training, as it also serves as a first approach to prepare monkeys for specific research tasks.

Current design of the task

The final version of the task consists of a grid of gray squares that is presented for a random amount of time, following which a target square with a higher contrast is displayed. The goal of the task is for the target to be touched by the monkey once it appears. The main parameter that distinguishes a distractor (gray square) from the target is its contrast. This task can be classified as a contrast discrimination task, aligning it with tasks that are performed in our lab for visual impairment pre-translational research.

The flowchart below represents the flow of the task, and there are 4 possible outcomes : 1) The target is not touched in time (2 seconds by default), so the trial is considered as *ignored*. 2) Something else other than the target is touched by the monkey, so the trial is deemed *incorrect*. 3) If the target is touched in time, then the trial is considered *correct*, and the schedule is assessed. 4) If the schedule criteria (i.e. ratio or interval) are met, then a *reward* (juice) is given to the monkey.

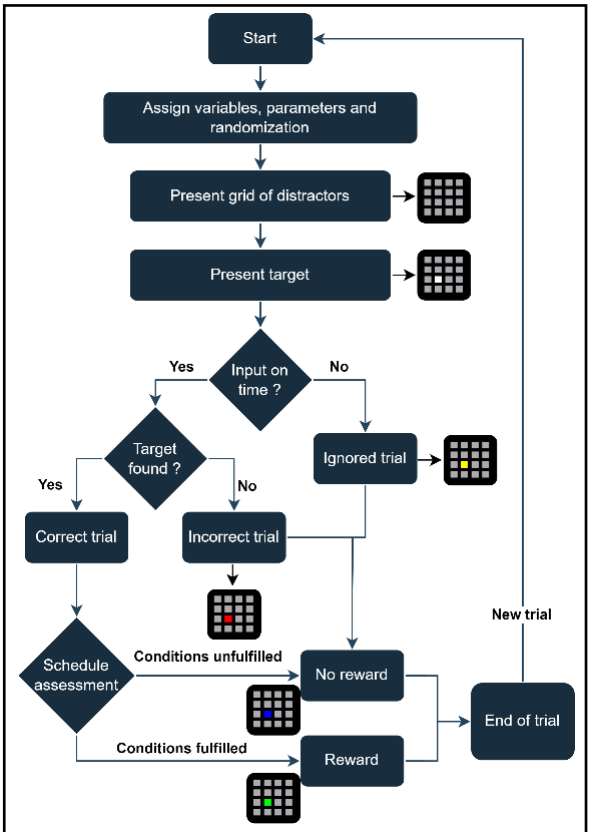

Figure 9 : Flowchart of the current task design.

Data analysis

Datasets (MWK2 files) are produced upon the conclusion of each session as the XBI tasks run through MWorks. The datasets contain all information through all frames that were displayed on the XBI, to analyze them, a MATLAB routine was created that runs through every stimulus presented every time the XBI display is updated, and uses them to recreate the timeline of the session.

With this routine, information on the outcomes of every trial is obtained, and a plot for the timeline and cumulative reward of the session can thus be made.

Pre-training protocol

A lot of room for customization is left by this grid task design as most of its

parameters are easily modifiable: Grid size, stimuli size, reward schedules, contrasts, and so on. In order to ensure that the pre-training is effective across multiple monkeys, a protocol needs to be designed that will train each monkey across multiple difficulty levels depending on their *performance*, *performance stability*, and *engagement*.

During this study, the outline of the protocol was designed, which is composed of 5 phases :

Phase 0 has the entirety of the XBI screen as the distractor and target, and is associated with continuous reinforcement, so a reward is obtained by the monkey when it touches the screen. Its purpose is for the monkey to learn to interact with the XBI.

Phase 1 presents one big square at the center of the screen, which becomes the target, and is also associated with continuous reinforcement. In this phase, there is a possibility of error, as an incorrect trial occurs when the screen is pressed outside of the target. The aim is to habituate the monkey to touching squares and introduce the concept of error.

Phases 2 to 4 follow a similar concept, where the number of squares in the grid is increased (2x2 → 4x4 → 6x6) to enhance the task difficulty. Phase 2 will initially be associated with fixed ratio or interval schedules, and once the monkeys become familiar with these schedules, variability is introduced.

In this project, the main focus will be on phases 0 to 2, as phases 3 and 4 can be difficult even for humans depending on the contrast and presentation delay of the target. They will be reserved for cases where the task becomes too easy for the monkeys after long-term interaction and will mainly be associated with variable reward schedules.

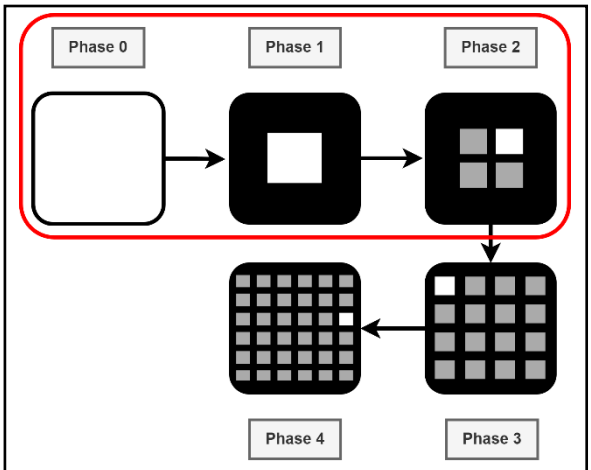

Figure 10 : Illustration of the current protocol outline with a focus on phases 0 to 2.

Results and observations

Experiment setup

For the pre-training sessions it is planned to implement the XBI with the task inside the main enclosure of the monkeys. The AI will be available through an opening in the border of the enclosure.

However, for this specific project, we only tested the task and observed the interaction of the monkeys with it, so a specific monkey, Stan, was placed inside an individual cage with access to the XBI.

Each session was 50 minutes to 1h30 long and recorded with the first one being recorded by outside cameras to observe the behavior of the monkey. The rewards given during the task are 0.1 mL of juice.

Observations in Phase 0

We had a first interaction between Stan, a trained monkey from our lab, and the XBI task. We presented the phase 0 of the protocol to Stan for a session of 1h20 using continuous reinforcement.

The result was quite straightforward, the first trials were ignored, most likely because the monkey was still figuring out what was happening. After 4 minutes we have the first correct trials after which many follow. In all we had 1531 trials, 731 of

which were correct and rewarded while 800 were ignored.

In the timeline we find multiple periods with a high concentration of correct trials, the main ones being from around 12 to 22 minutes and 27 to 40 minutes. While observing the recordings we noticed that Stan was pressing his back against the XBI screen and so rewards were being continuously delivered. Although being a bypass of the task rules, this was accounted for and considered as a success since it means that the monkey associated touching the XBI with being given rewards.

As for the cumulative rewards, the session ended with a total of 731 rewards of 0.1 mL of juice, so a total of 73.1 mL of juice were given.

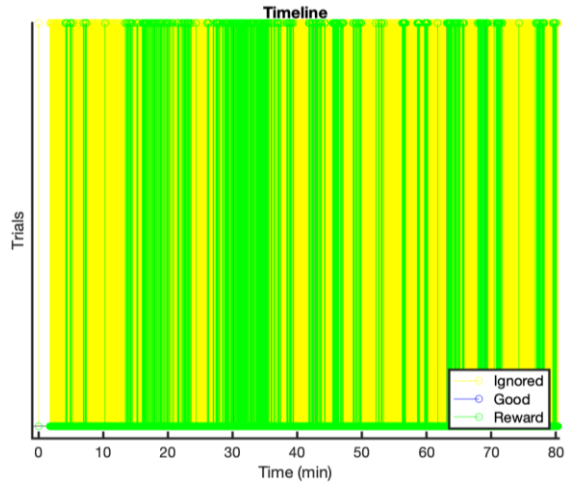

Figure 11 : Stem plot of the timeline for the session in phase 0 with the type of trials in the vertical axis and the time in minutes in the horizontal axis. Trials are represented in yellow for ignored, green for correct and rewarded. Correct trials are hidden by the rewards as they superpose each other.

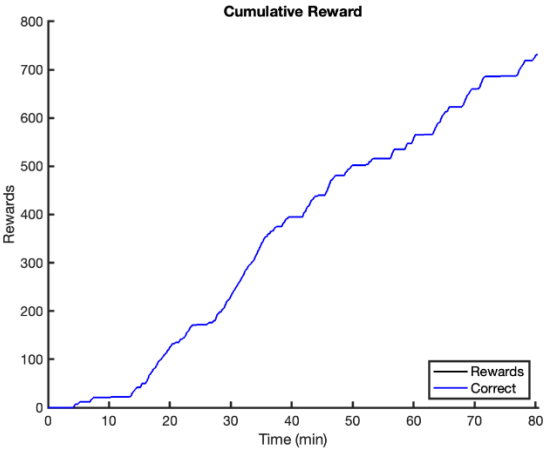

Figure 12 : Cumulative reward plot for the session in phase 0 with the cumulative number of rewards on the vertical axis and the time of the session on the horizontal axis. Again the rewards and correct trials are superposed as we use continuous reinforcement.

Observations in Phase 1

Considering phase 0 as a success for our testing session, we initialized a phase 1 session with the same monkey but decided to remove the incorrect trials as we suspected that it could negatively impact the association of touching the target with receiving a reward.

Here we clearly noticed a high decrease in engagement from Stan as from a session of 56 minutes with a total of 811 trials only 13 were correct and 798 were labeled as ignored.

Since in phase 1 there is a single square at the center of the screen that has to be touched, the strategy of pressing the back against the screen stopped functioning which explains the decrease in correct trials. Sadly, this session wasn't video recorded so we cannot confirm if the monkey continued trying its strategy and so the correct trials observed were due to chance (by randomly touching the target with his back) or if he adapted the technique and so the correct trials are due to understanding the task. Although considering the very small number of good trials the former is more likely.

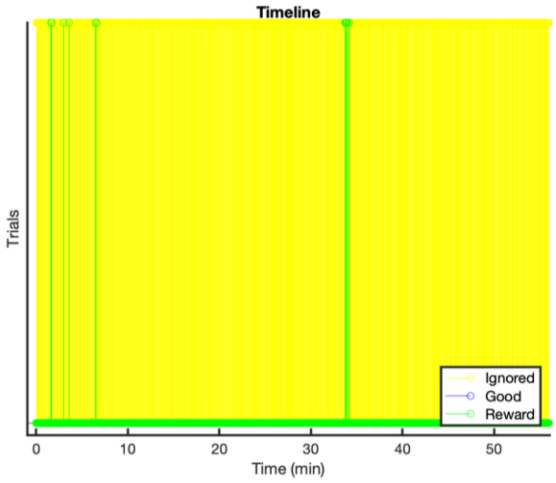

Figure 13 : Stem plot of the timeline for the session in phase 1.

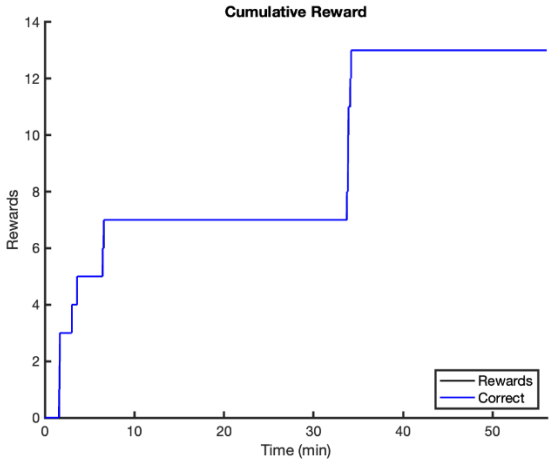

Figure 14 : Cumulative reward plot for the session in phase 1.

Rundown of the two first sessions

These two first sessions with the XBI led to 3 main discoveries concerning the task. *First*, many coding optimizations were needed for the task itself and the MATLAB routine, mostly to correct bugs that weren't identifiable before real tests. *Second*, the difficulty curve between the phases 0 and 1 seemed too high and so motivated the removal of incorrect trials. *Third*, the labeling of ignored trials isn't specific enough because the time window to touch the target was too short (2 seconds). This means that ignored trials can actually be "not fast enough" trials and so they don't characterize engagement as they should.

Discussion

The goals of this project were : **1)** To design an engaging touchscreen task that functions in tandem with the XBI. **2)** Prepare a MATLAB routine that automatically analyses the dataset produced by the task. **3)** Design a protocol for standardization of the task across multiple monkeys. **4)** Study the effects of variable reward schedules on engagement. **5)** Determine if pre-training increases welfare and decreases overall experiment time.

As showed in the results, phase 0 producing such great engagement during Stan's first session can be counted as a success for the touchscreen task design. The MATLAB routine is functional as demonstrated with figures 11 to 14 which were made using the routine. The protocol is still in its early stages of development as we could observe a clear engagement decrease during the session in phase 1. As for the study of variable reward schedules effects and the role of pre-training on welfare and overall experiment efficiency, these have not been demonstrated during this project.

As a preliminary study focused on the development of a specific touchscreen task, the experimental part is in its early stages. The achievements attained during this project were :

- The designing and coding in MWEL of the task.
- The implementation of reward schedules (ratio and interval).
- Designing the outline of the protocol.
- Coding a MATLAB routine for the analysis of produced datasets.

We have confirmed the viability of the current task design during interaction with a trained monkey, as it induced a promising engagement level during phase 0, but we

would benefit from more observations in that regard.

The overall aims of the study being to increase wellbeing and training welfare, we cannot confirm any influence of our pre-training protocol in those goals.

The effect comparison of fixed reward schedules against variable reward schedules on engagement couldn't be investigated during this preliminary study.

We identified some issues with the parameters choice for the task design : The delays between the presentation of each stimulus were too long which made each trial longer than they needed to, so we shortened these delays. The target presentation delay was too short which decreased the relevancy of ignored trials, therefore we increased it to 10 seconds. We also suspect that the size of the single square in phase 1 might be too little for the subjects to be attracted to it so we reserve the possibility of increasing it.

As for the protocol, in this study we only designed its outline, the separation in multiple phases seems appropriate but since we modified phase 1 by removing the incorrect trials, we might need to add this as an intermediary between the current phases 0 and 1. The protocol criteria are still in development, we plan to adjust the phases according to 3 factors : *Performance* (correct trials / total trials – ignored trials), *performance stability* (how much the stability varies between each session, if it varies too much then we need to adjust the phase) and *engagement* which is a very important criteria to follow since we want to maximize the interactions between animals and the XBI during pre-training. However, engagement is not an easily quantifiable parameter as it depends on multiple factors, and we must pre-determine which values of engagement are acceptable.

In conclusion, we managed to establish the foundation of the XBI pre-training overall project during this preliminary study. The structure of the touchscreen task is complete, we can very easily adjust the reward schedules inside the task, and we have a routine that analyzes the datasets that are produced after each session. Although we have very little observational data, what we obtained seemed promising for the viability of the task.

Future perspectives

Due to the developmental nature of this study we will separate the perspectives into two parts : One where we will discuss what will come next for the project and the other where we will consider future applications of the XBI task setup.

Continuation of the project

As we couldn't have much observation of interactions between the monkeys and the XBI we're missing information necessary to complete the project. As such, the next stages will be focused on doing multiple sessions of XBI training with different monkeys to accomplish 4 main goals : 1) Resolving possible issues observed in the task. 2) Experiment the setup with naïve monkeys to adjust the difficulty curve. 3) Complete the protocol phases and the criteria for phase adjustment. 4) Test the 3 hypotheses raised during this project.

Future applications

Since the XBI is a very customizable tool, it provides a vast number of possibilities for research application. Moreover, as this study is based on unsupervised tasks, we can apply it in parallel to the main time frame of research without the risk of disturbing it. As such, one possible application for our laboratory would be to

implement more specific tasks, designed for data collection and pre-translational research, into the XBI setup. Our lab, for instance, has a long-term goal of studying possible associations of optogenetics with visual prosthesis as a method to treat visual impairment. For this situation we could envision the integration of future experiments with this unsupervised setup for a continuous collection of data and performance improvement with the prosthesis.

This project can also be standardized for other studies and research groups, but the task would need to be redesigned specifically for their needs. What would then be transmitted from our future discoveries is the reward schedule progression and the choice of criteria for the protocol adjustment.

Acknowledgements

I would like to thank the Schmid research group and my two tutors Jaime Daniel Cadena Valencia and Marcus Haag for their support and guidance throughout this project which couldn't have been completed without their assistance.

References

"APA Dictionary of Psychology." n.d. Accessed May 28, 2023. https://dictionary.apa.org/.

Berger, M., A. Calapai, V. Stephan, M. Niessing, L. Burchardt, A. Gail, and S. Treue. 2018. "Standardized Automated Training of Rhesus Monkeys for Neuroscience Research in Their Housing Environment." *Journal of Neurophysiology* 119 (3): 796–807. https://doi.org/10.1152/jn.00614.2017.

Calapai, A., M. Berger, M. Niessing, K. Heisig, R. Brockhausen, S. Treue, and A. Gail. 2017. "A Cage-Based Training, Cognitive Testing and Enrichment System Optimized for Rhesus Macaques in Neuroscience Research." *Behavior Research Methods* 49 (1): 35–45. https://doi.org/10.3758/s13428-016-0707-3.

"Experiment Language (MWEL) — MWorks 0.13.Dev Documentation." n.d. Accessed May 29, 2023. https://mworks.github.io/documentation/latest/mwel/index.html.

Hofman, Michel. 2014. "Evolution of the Human Brain: When Bigger Is Better." *Frontiers in Neuroanatomy* 8. https://www.frontiersin.org/articles/10.3389/fnana.2014.00015.

Kiorpes, Lynne. 2016. "The Puzzle of Visual Development: Behavior and Neural Limits." *Journal of Neuroscience* 36 (45): 11384–93. https://doi.org/10.1523/JNEUROSCI.2937-16.2016.

McMillan, Jennifer L, Mollie A Bloomsmith, and Mark J Prescott. 2017. "An International Survey of Approaches to Chair Restraint of Nonhuman Primates." *Comparative Medicine* 67 (5): 442–51.

Skinner, B. F., and C. B. Ferster. 2015. *Schedules of Reinforcement*. B. F. Skinner Foundation.

Staddon, J. E. R., and D. T. Cerutti. 2003. "Operant Conditioning." *Annual Review of Psychology* 54 (1): 115–44. https://doi.org/10.1146/annurev.psych.54.101601.145124.

"The MWorks Project." n.d. Accessed May 29, 2023. https://mworks.github.io/.

Appendices

Appendix 1 : MWEL script for launching the touchscreen task

```
////////////////////////////////////////////////////////////////////////////
//
// Platform-specific I/O
//
////////////////////////////////////////////////////////////////////////////

var touch_in_progress = false


iodevice/mio pointer (
    data_interval = 1ms
    reward_a = IO_rewardA
    reward_b = IO_rewardB
    touch_x_calib = pointer_x
    touch_y_calib = pointer_y
    )


%define reset_pointer_position (x_position, y_position)
   assert (
      condition = not touch_in_progress
      message = 'Subject is touching the display too early'
      )
   pointer_x = x_position
   pointer_y = y_position
%end


////////////////////////////////////////////////////////////////////////////
//
// The remainder of the experiment is shared among all platforms
//
////////////////////////////////////////////////////////////////////////////


%include gridXBI2
```

Appendix 2 : MWEL script for the parameters and protocol of the task

```
//////////////
//Variables//
//////////////
group protocol_variable {
   var protocol_type = 2 //2 = each Xth / 3 = X delay
   var grid_type = 0 // 0 = whole screen / 1 = 1x1 / 2 = 2x2 / 3 = 4x4 / 4 = 6x6
   var ratio_min_max = [1,1] //ratio needed to get reward on protocol 2
   var interval_min_max_s = [5,5] //duration of the delay before getting reward
   var good_delay_ms = 300 //duration of the good trial screen
   var bad_delay_ms = 1000 //duration of the bad trial screen
   var off_delay_ms = 500 //delay before showing a new grid
   var time_to_find_ms = 10000 //time to find the target before trial is considered as ignored
   var make_distractor_zone = 0 //if = 0 then it deactivates the bad trials
   var IO_rewardA = 0 //creating variable for how many seconds the pump stays open (for liquid reward)
   var IO_rewardB = 0 //we also need to create the variable for pump B, else it doesn't launch
   var reward_val = .1 //this is the amount of liquid that the monkey gets per reward
}
group variable_stimuli {
   var x_values = [-15, -9, -3, 3, 9, 15] //x and y coordinates for the position of the stimuli
   var square_size = 3.5 //size of the squares in grid 6x6
   var square_size4x4 = 4 //size of the squares in grid 4x4
   var square_size2x2 = 4.5 //size of the squares in grid 2x2
   var center_size = 8 //size of the single square in grid 1x1
   var target_size = 0 //size of the target and responses --> placeholder
}
group variable_target {
   var pointer_x = 5000
   var pointer_y = 5000
   var pointer_on_target = False
   var pointer_on_distractor = False
   var contr_t = 1 //color for target --> contrast
   var contr_d = 0.35 //color for distractor --> contrast
   var x_pos = 0
   var y_pos = 0
}
group protocol_fixed {
   var ratio_counter = 0 //counter for the ratio protocol
   var variable_interval_s = 0
   var variable_ratio = 0
   var distractor_size = 1000 //size of the distractor field and whole screen target
   var grid_delay_ms = 0
   var grid_delay_values_ms = [1, 10] //minimum and maximum for presentation delay (is multiplied by 100)
}
group follow_variables { //grouping the variables that will be stored during trials
   var trial_number = 0
   var good_trials = 0
   var bad_trials = 0
```

```
    var ignored_trials = 0
    var rewards_gotten = 0
}

//%require reset_pointer_position

//////////////
//My stimuli//
//////////////
stimulus_group distractors { //this makes a group of stimuli to store the results of the loops of distractors
    range_replicator ( //loop that makes all stimuli for the first row
        variable = index //the variable that differentiates each stimulus made
        from = 0 //from which value we start counting
        to = 5 //until which value we count
        step = 1 //we count by steps of one so : 1 --> 2 --> 3 --> ...
    ) {
        rectangle distractor1${index} ( //here are the parameters of the stimuli
            color = contr_d,contr_d,contr_d //contrast of the stimuli
            x_size = square_size //size of the stimuli --> modified later depending on the grid type
            x_position = x_values[${index}] //this ties the column of the stimulus with it's index
            y_position = x_values[5] //same y value for every one because they're on the same row
        )
    }
    range_replicator ( //loop that makes all stimuli for the second row
        variable = index
        from = 0
        to = 5
        step = 1
    ) {
        rectangle distractor2${index} (
            color = contr_d,contr_d,contr_d
            x_size = square_size
            x_position = x_values[${index}]
            y_position = x_values[4]
        )
    }
    range_replicator ( //loop that makes all stimuli for the third row
        variable = index
        from = 0
        to = 5
        step = 1
    ) {
        rectangle distractor3${index} (
            color = contr_d,contr_d,contr_d
            x_size = square_size
            x_position = x_values[${index}]
            y_position = x_values[3]
        )
    }
```

```
    range_replicator ( //loop that makes all stimuli for the fourth row
        variable = index
        from = 0
        to = 5
        step = 1
    ) {
        rectangle distractor4${index} (
            color = contr_d,contr_d,contr_d
            x_size = square_size
            x_position = x_values[${index}]
            y_position = x_values[2]
        )
    }
    range_replicator ( //loop that makes all stimuli for the fifth row
        variable = index
        from = 0
        to = 5
        step = 1
    ) {
        rectangle distractor5${index} (
            color = contr_d,contr_d,contr_d
            x_size = square_size
            x_position = x_values[${index}]
            y_position = x_values[1]
        )
    }
    range_replicator ( //loop that makes all stimuli for the sixth row
        variable = index
        from = 0
        to = 5
        step = 1
    ) {
        rectangle distractor6${index} (
            color = contr_d,contr_d,contr_d
            x_size = square_size
            x_position = x_values[${index}]
            y_position = x_values[0]
        )
    }
}
rectangle presentation_marker ( //this marks the presentation state for the data analysis
    color = 0,0,0
    x_size = 1
    x_position = -20
    y_position = -20
)

rectangle center_distractor ( //here we make the big distractor positioned at the center --> for
1x1 grid
    color = contr_d,contr_d,contr_d
```

```
    x_size = center_size
    x_position = 0
    y_position = 0
)

fixation_point target_square ( //here we make the fixation point that will trigger when we touch
it
    color = contr_t,contr_t,contr_t
    x_size = target_size
    x_position = x_pos
    y_position = y_pos
    trigger_width = target_size
    trigger_watch_x = pointer_x
    trigger_watch_y = pointer_y
    trigger_flag = pointer_on_target
)
fixation_point distractor_zone ( //here we make a zone that makes it so that when we touch at
the wrong place we get a wrong answer
    color = 0,0,0
    x_size = distractor_size //covers the whole screen
    x_position = 0
    y_position = 0
    trigger_width = distractor_size
    trigger_watch_x = pointer_x
    trigger_watch_y = pointer_y
    trigger_flag = pointer_on_distractor
)
blank_screen training_screen ( //whole screen becomes the distractor --> phase 0 (training)
    color = contr_d,contr_d,contr_d
)

%define square_answer (color) //this is a function to produce rectangles with different colors -
-> indicate result of trial (good, bad, reward, ignored)
    rectangle (
        color = color
        x_size = target_size
        x_position = x_pos
        y_position = y_pos
    )
%end
//here we call the function to make 4 stimuli with different colors
square_answer good_answer (
    color = 0,0,1 //blue
)
square_answer bad_answer (
    color = 1,0,0 //red
)
square_answer reward (
    color = 0,1,0 //green
)
```

```
square_answer ignored (
   color = 1,1,0 //yellow
)

//////////
// Macros//
//////////
%define queue_grid() //this function queues the stimuli needed depending on the type of grid we want
if (grid_type == 4) { //6x6 grid so we have 6 rows with 6 stimuli each (6 columns)
   queue_stimulus(distractor10)
   queue_stimulus(distractor11)
   queue_stimulus(distractor12)
   queue_stimulus(distractor13)
   queue_stimulus(distractor14)
   queue_stimulus(distractor15)

   queue_stimulus(distractor20)
   queue_stimulus(distractor21)
   queue_stimulus(distractor22)
   queue_stimulus(distractor23)
   queue_stimulus(distractor24)
   queue_stimulus(distractor25)

   queue_stimulus(distractor30)
   queue_stimulus(distractor31)
   queue_stimulus(distractor32)
   queue_stimulus(distractor33)
   queue_stimulus(distractor34)
   queue_stimulus(distractor35)

   queue_stimulus(distractor40)
   queue_stimulus(distractor41)
   queue_stimulus(distractor42)
   queue_stimulus(distractor43)
   queue_stimulus(distractor44)
   queue_stimulus(distractor45)

   queue_stimulus(distractor50)
   queue_stimulus(distractor51)
   queue_stimulus(distractor52)
   queue_stimulus(distractor53)
   queue_stimulus(distractor54)
   queue_stimulus(distractor55)

   queue_stimulus(distractor60)
   queue_stimulus(distractor61)
   queue_stimulus(distractor62)
   queue_stimulus(distractor63)
   queue_stimulus(distractor64)
```

```
    queue_stimulus(distractor65)
}
if (grid_type == 3) { //4x4 grid --> 4 rows and 4 columns
    square_size = square_size4x4
    queue_stimulus(distractor21)
    queue_stimulus(distractor22)
    queue_stimulus(distractor23)
    queue_stimulus(distractor24)

    queue_stimulus(distractor31)
    queue_stimulus(distractor32)
    queue_stimulus(distractor33)
    queue_stimulus(distractor34)

    queue_stimulus(distractor41)
    queue_stimulus(distractor42)
    queue_stimulus(distractor43)
    queue_stimulus(distractor44)

    queue_stimulus(distractor51)
    queue_stimulus(distractor52)
    queue_stimulus(distractor53)
    queue_stimulus(distractor54)
}
if (grid_type == 2) { //2x2 grid --> 2 rows and 2 columns
    square_size = square_size2x2
    queue_stimulus(distractor32)
    queue_stimulus(distractor33)

    queue_stimulus(distractor42)
    queue_stimulus(distractor43)
}
if (grid_type == 1) { //queueing the big center square for the phase 1 of training
    queue_stimulus(center_distractor)
}
if (grid_type == 0) { //queueing the big screen for the phase 0 of training
    queue_stimulus(training_screen)
}
%end

%define assign_size() //caution : grid_type has to be changed when game is stopped, the stimuli
don't update their size in real time!!!!
    if (grid_type == 0) { //here we make the target really big because it needs to cover the whole
screen
        target_size = distractor_size
    }
    if (grid_type == 1) { //if grid_type = 1 then the size of the target and response stimuli is the
same as center_distractor --> so it's big
        target_size = center_size
    }
```

```
   if (grid_type == 2) { //makes the size of the target the same as the squares from grid 2x2
      target_size = square_size2x2
   }
   if (grid_type == 3) { //makes the size of the target the same as the squares from grid 4x4
      target_size = square_size4x4
   }
   if (grid_type == 4) { //makes the size of the target the same as the squares from grid 6x6
      target_size = square_size
   }
%end

%define random_position() //setting the possible positions of the target depending on the grid
if (grid_type == 4) {
   x_pos = x_values[disc_rand(0, 5)]
   y_pos = x_values[disc_rand(0, 5)]
}
if (grid_type == 3) {
   x_pos = x_values[disc_rand(1, 4)]
   y_pos = x_values[disc_rand(1, 4)]
}
if (grid_type == 2) {
   x_pos = x_values[disc_rand(2, 3)]
   y_pos = x_values[disc_rand(2, 3)]
}
if (grid_type == 1 or grid_type == 0) {
   x_pos = 0
   y_pos = 0
}
%end

%define var_rand() //this function randomizes the ratio and interval for the reward schedules
   variable_interval_s = disc_rand(interval_min_max_s[0], interval_min_max_s[1])
   variable_ratio = disc_rand(ratio_min_max[0], ratio_min_max[1])
%end

%define random_delay() //this function randomizes the presentation delay of the grid --> time
where we have to wait for the target to appear
   grid_delay_ms = disc_rand(grid_delay_values_ms[0],grid_delay_values_ms[1]) * 100
//random value (min and max chosen from the list) that is then multiplied by 100
%end

%define randomize_values()
   random_position() //creating a random value for the position (x,y) of the target
   random_delay() //creating a random value for the grid presentation delay
%end

%define choose_protocol_type () //this function assigns the different schedules depending on
the type of protocol chosen
   if (protocol_type == 2) { //if protocol_type = 2 --> assign the parameters for ratio schedule
      ratio_counter = 0 //setting the counter for ratio at 0
```

```
      variable_ratio = variable_ratio
   }
   if (protocol_type == 3) { //if protocol_type = 3 --> assign the parameters for interval schedule
      variable_interval_s = variable_interval_s //duration of the delay before reward
      start_timer ( //starting timer for schedule delay
         timer = variable_timer
         duration = variable_interval_s
         duration_units = s
      )
   }
%end

%define good_answer_poss() //defines what happens in the good_answer state depending on protocol
   if (protocol_type == 2) {
      ratio_counter += 1 //adding one to the ratio counter
   }
%end

%define timer(x) //creating a function to easily make timers that differ only in duration
   start_timer(
      timer = local_timer
      duration = x
      duration_units = ms
   )
%end

%define is_distractor_zone() //function that adds the distractor zone if the variable = 1 --> if it's = to 0 then impossible to get bad trials even if we press the whole screen
   if (make_distractor_zone == 1) {
      queue_stimulus(distractor_zone)
   }
%end

%define grid_presentation() //function that presents the grid and sets a timer before the target appears
   queue_grid()
   queue_stimulus(presentation_marker)
   update_display()
   timer(grid_delay_ms) //this timer determines for how long the grid is just presented without target
%end

%define starting_task()
   reset_pointer_position( //resetting the position of the pointer
      x_position = 5000
      y_position = 5000
   )
   is_distractor_zone()
   dequeue_stimulus(presentation_marker)
```

```
   queue_grid()
   queue_stimulus(target_square) //this is the target stimuli (with a different contrast)
   update_display()
   timer(time_to_find_ms)
%end

////////////
//protocol//
////////////

stimulus_display ( //sets the background of the experiment (it's colored in black to focus on the stimuli)
   background_color = 0,0,0
)

protocol 'All in one' {
   start_io_device(pointer)

   task 'first task' {

      state 'assign variability' { //this state assigns the type of protocol, size of stimuli and more importantly the randomization of delay and ratio
         var_rand()              //we only get back to this state when we get the reward in protocol type 1 or 2 --> since that's the only moment we want to reset the ratio or delay
         choose_protocol_type()     //it also resets the ratio counter in addition to the randomization of delay and ratio
         assign_size()
         goto ('assign parameters')
      }
      state 'assign parameters' { //here we assign the parameters of the grid
         trial_number += 1
         randomize_values()
         goto('presenting grid')
      }
      state 'presenting grid' {
         grid_presentation()
         goto (
            target = 'start task'
            when = timer_expired(local_timer)
         )
      }
      state 'start task' { //here we queue all the stimuli of the grid
         starting_task()
         goto (
            target = 'ignored'
            when = timer_expired(local_timer)
         )
         goto (
            target = 'good answer'
            when = pointer_on_target
```

```
    )
    goto (
        target = 'bad answer'
        when = pointer_on_distractor and make_distractor_zone == 1
    )
}
state 'good answer' { //if we find the target then we get here
    good_trials += 1
    dequeue_stimulus(target_square)
    queue_stimulus(good_answer) //queueing the good stimulus
    good_answer_poss() //increases the ratio counter by one if in protocol 2
    update_display()
    timer(good_delay_ms)
    goto (
        target = 'reward'
        when = ratio_counter == variable_ratio and protocol_type == 2
    )
    goto (
        target = 'reward'
        when = timer_expired(variable_timer) and protocol_type == 3
    )
    goto (
        target = 'off'
        when = timer_expired(local_timer)
    )
}
state 'reward' {
    rewards_gotten += 1
    dequeue_stimulus(good_answer)
    queue_stimulus(reward)
    update_display()
    IO_rewardA = reward_val

    timer(good_delay_ms)
    goto (
        target = 'off2'
        when = timer_expired(local_timer)
    )
}
state 'ignored' { //if we don't find the target in time we get here
    ignored_trials += 1
    dequeue_stimulus(target_square)
    queue_stimulus(ignored) //queueing the ignored stimulus
    update_display()
    timer(bad_delay_ms)
    goto (
        target = 'off'
        when = timer_expired(local_timer)
    )
}
```

```
        state 'bad answer' {
            bad_trials += 1
            dequeue_stimulus(target_square)
            queue_stimulus(bad_answer)
            update_display()
            timer(bad_delay_ms)
            goto (
                target = 'off'
                when = timer_expired(local_timer)
            )
        }
        state 'off' { //this state clears the display and makes a delay before starting next task
            clear_display()
            timer(off_delay_ms)
            goto (
                target = 'assign parameters' //have to go back to assign parameters to randomize target
position
                when = timer_expired(local_timer)
            )
        }
        state 'off2' {
            clear_display()
            timer(off_delay_ms)
            goto (
                target = 'assign variability'
                when = timer_expired(local_timer)
            )
        }
    }
}
```

Appendix 3 : MATLAB routine for analysis of MWK2 files

```
clc
clear

%%%%%%
%Locating and opening the dataset / also getting the scripts for analysis
addpath('/Library/Application Support/MWorks/Scripting/MATLAB/')
filename = '/Users/diogorocha/Documents/MWorks/Data/datamonkey1.mwk2';

%%%%%%
%Getting the events from the dataset
r_codec = getReverseCodec(filename);
upd = r_codec('#stimDisplayUpdate');
upd_evs = getEvents(filename, upd);
prt = r_codec('protocol_type'); %finds the type of protocol
prt_ratio = r_codec('ratio_min_max'); %find the ratio
prt_interval = r_codec('interval_min_max_s'); %find the inverval
grid = r_codec('grid_type'); %find the type of grid
error = r_codec('make_distractor_zone');

session_var.protocol = getEvents(filename,prt);
session_var.grid = getEvents(filename,grid);
session_var.ratio = getEvents(filename,prt_ratio);
session_var.interval = getEvents(filename,prt_interval);
session_var.error = getEvents(filename,error);

%%%%%%
%Find phase and type of schedule
phase_i = session_var(1).grid.data; %determines which phase was used
for i=0:4
   if phase_i == i
      phase = "Phase " + i;
   end
end
%Find if there was a distractor zone (=1) or not (=0)
dis_zone = session_var(1).error.data; %stores the value of make_distractor_zone

schd = session_var(1).protocol.data; %determines which schedule was used
if schd == 2
   schd = "Ratio schedule";
   ratio = session_var(1).ratio(1).data;
elseif schd == 3
   schd = "Interval schedule";
   interval = session_var(1).interval(1).data;
end

%%%%%%
%Organizing timestamps and stimuli
target_squares = struct('Timestamp', {}, 'Stimulus', {}); %{} as a way to initialize str with
empty arrays
```

```
good_answers = struct('Timestamp', {}, 'Stimulus', {});
rewards = struct('Timestamp', {}, 'Stimulus', {});
if ~phase_i == 0 && ~dis_zone == 0
bad_answers = struct('Timestamp', {}, 'Stimulus', {});
end
ignored = struct('Timestamp', {}, 'Stimulus', {});
for upd_index = 1:length(upd_evs)
   for stim_index = 1:length(upd_evs(upd_index).data)
      if(length(upd_evs(upd_index).data) > 1)
         %comparing the name of the stimulus with target_square to take
         %only the target_square stimuli --> they represent trial start
         if(strcmp('target_square',upd_evs(upd_index).data{stim_index}.name))
            %getting the timestamp of each stimuli and subtracting by
            %the initial timestamp to set the first display update as 0
            target_squares(end+1).Timestamp = timeconv(upd_evs(upd_index).time_us -
upd_evs(1).time_us);
            %getting the structure of the stimulus
            target_squares(end).Stimulus = upd_evs(upd_index).data{stim_index};
         end
         if(strcmp('good_answer',upd_evs(upd_index).data{stim_index}.name))
            good_answers(end+1).Timestamp = timeconv(upd_evs(upd_index).time_us -
upd_evs(1).time_us);
            good_answers(end).Stimulus = upd_evs(upd_index).data{stim_index};
         end
         if(strcmp('reward',upd_evs(upd_index).data{stim_index}.name))
            rewards(end+1).Timestamp = timeconv(upd_evs(upd_index).time_us -
upd_evs(1).time_us);
            rewards(end).Stimulus = upd_evs(upd_index).data{stim_index};
         end
         if(strcmp('bad_answer',upd_evs(upd_index).data{stim_index}.name))
            bad_answers(end+1).Timestamp = timeconv(upd_evs(upd_index).time_us -
upd_evs(1).time_us);
            bad_answers(end).Stimulus = upd_evs(upd_index).data{stim_index};
         end
         if(strcmp('ignored',upd_evs(upd_index).data{stim_index}.name))
            ignored(end+1).Timestamp = timeconv(upd_evs(upd_index).time_us -
upd_evs(1).time_us);
            ignored(end).Stimulus = upd_evs(upd_index).data{stim_index};
         end
      end
   end
end

%%%%%%
%Putting everything into one big structure
%The composition of the structure is different depending on which type of
%schedule and phase is being used
if schd == "Ratio schedule"
   if phase == "Phase 0" || dis_zone == 0
      all_data = struct( ...
```

```
        'Names',{"Trials","Good","Rewards","Ignored"}, ...
        'Stimuli',{target_squares,good_answers,rewards,ignored}, ...
        'Variables',{"Phase","Schedule","Ratio_min_max","Distractor_zone"}, ...
        'Values',{phase,schd,[ratio{1},ratio{2}],dis_zone});
    else
        all_data = struct( ...
        'Names',{"Trials","Good","Rewards","Bad","Ignored"}, ...
        'Stimuli',{target_squares,good_answers,rewards,bad_answers,ignored}, ...
        'Variables',{"Phase","Schedule","Ratio_min","Ratio_max","Distractor_zone"}, ...
        'Values',{phase,schd,ratio{1},ratio{2},dis_zone});
    end
elseif schd == "Interval schedule"
    if phase == "Phase 0" || dis_zone == 0
        all_data = struct( ...
        'Names',{"Trials","Good","Rewards","Ignored"}, ...
        'Stimuli',{target_squares,good_answers,rewards,ignored}, ...
        'Variables',{"Phase","Schedule","Interval_min_max","Distractor_zone"}, ...
        'Values',{phase,schd,[interval{1},interval{2}],dis_zone});
    else
        all_data = struct( ...
        'Names',{"Trials","Good","Rewards","Bad","Ignored"}, ...
        'Stimuli',{target_squares,good_answers,rewards,bad_answers,ignored}, ...
        'Variables',{"Phase","Schedule","Interval_min","Interval_max","Distractor_zone"}, ...
        'Values',{phase,schd,interval{1},interval{2},dis_zone});
    end
end

%%%%%
%Making the structure with trial outcomes
all_outcomes = outcome_rec(upd_evs);

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%making an array that contains all timestamps from the experiment converted
%to milliseconds and subtracted by the first timestamp to set it as t0
%had to make the conversion directly in the loop to avoid rounding issues
all_timestamps = [];
for i=1:length(upd_evs)
    all_timestamps(end+1) = timeconv(upd_evs(i).time_us - upd_evs(1).time_us);
end

%%%%%%
%here I compare all the timestamps with those from the target_square
%stimuli. For ismember to work I had to convert the field of timestamps
%from all_data into an array by putting it inside [...]
%target_vals is composed of zeros and ones (0 = false and 1 = true)
if ~phase_i == 0 && ~dis_zone == 0
    target_vals = ismember(all_timestamps, [all_data(1).Stimuli.Timestamp]);
    good_vals = ismember(all_timestamps, [all_data(2).Stimuli.Timestamp]);
    reward_vals = ismember(all_timestamps, [all_data(3).Stimuli.Timestamp]);
```

```
    bad_vals = ismember(all_timestamps, [all_data(4).Stimuli.Timestamp]);
    ignored_vals = ismember(all_timestamps, [all_data(5).Stimuli.Timestamp]);
elseif phase_i == 0 || dis_zone == 0
    target_vals = ismember(all_timestamps, [all_data(1).Stimuli.Timestamp]);
    good_vals = ismember(all_timestamps, [all_data(2).Stimuli.Timestamp]);
    reward_vals = ismember(all_timestamps, [all_data(3).Stimuli.Timestamp]);
    ignored_vals = ismember(all_timestamps, [all_data(4).Stimuli.Timestamp]);
end

%%%%%%
%Call cumulative plot function
cml_rwd = cml_dist(reward_vals);
cml_ign = cml_dist(ignored_vals);
if ~phase_i == 0 && ~dis_zone == 0
cml_bad = cml_dist(bad_vals);
end
cml_good = cml_dist(good_vals);

%%%%%%
%Making the structure with all calculations
%Calculating : performance, interest,
Performance = length(good_answers) / (length(target_squares)-length(ignored)) * 100;
Interest = (length(target_squares)-length(ignored)) / length(all_timestamps) * 100;
all_results = struct('Calculations',{"Performance (%)","Interest (%)"}, ...
    'Results', {Performance,Interest});

%%%%%%
%Plotting (uncomment to plot, comment to ignore the plot)
%converting the timestamps into minutes
timestamps_min = timeconv_min(all_timestamps); %array with timestamps converted in
minutes
%%%%%Obsolete plots%%%%%%%%
if phase_i == 0 || dis_zone == 0
    plottingp0(timestamps_min,ignored_vals,good_vals,reward_vals)
else
    plotting(timestamps_min,ignored_vals,good_vals,reward_vals,bad_vals)
end
%%%%%%Obsolete plots%%%%%%%%
%%%%%Definitive plots%%%%%%%
% if phase_i == 0 || dis_zone == 0
%     cml_plot0(timestamps_min,cml_rwd,cml_good)
% else
%     cml_plot(timestamps_min,cml_rwd,cml_good,cml_bad)
% end
%%%%%%Definitive plots%%%%%%

%%%%%%%%%%%%%%%%%%%%%%
%%%%Cumulative plot function --> uses same time axis as the other plots
%%%%%%%%%%%%%%%%%%%%%%
function output = cml_dist(x)
```

```
    output = [];
    y = 0;
    for i=1:length(x)
        if x(i) == 1
            y = y+1;
        end
        output(end+1) = y;
    end
end

%%%%%%%%%%%%%%%%%%%%%%%%
%%%%Plotting function%%%%
%%%%%%%%%%%%%%%%%%%%%%%%
function output = plotting(x,y1,y2,y3,y4)
    hold on
    % plot(all_timestamps,target_vals, color='black')
    stem(x,y1, color='yellow', LineWidth=1.25)
    stem(x,y2, color='blue', LineWidth=1.25)
    stem(x,y3, color='green', LineWidth=1.25)
    stem(x,y4, color='red', LineWidth=1.25)
    xlabel('Time (min)', 'FontSize', 14)
    ylabel('Trials', 'FontSize', 14)
    title('Timeline', 'FontSize', 16)
    legend('Ignored','Good','Reward','Bad', 'FontSize', 12,'Location','Southeast')
    set(gca, 'lineWidth', 2, 'FontSize', 12, 'YTick', [])
    xlim([-1 x(end)])
    ylim([-0.03 1])
    hold off
    saveas(gcf, '/Users/diogorocha/Documents/Plots/timeline.png', 'png')
end
function output = plottingp0(x,y1,y2,y3)
    hold on
    % plot(all_timestamps,target_vals, color='black')
    stem(x,y1, color='yellow')
    stem(x,y2, color='blue')
    stem(x,y3, color='green')
    xlabel('Time (min)', 'FontSize', 14)
    ylabel('Trials', 'FontSize', 14)
    title('Timeline', 'FontSize', 16)
    legend('Ignored','Good','Reward','FontSize',12,'Location','Southeast')
    set(gca, 'lineWidth', 2, 'FontSize', 12, 'YTick', [])
    xlim([-1 x(end)])
    ylim([-0.03 1])
    hold off
    saveas(gcf, '/Users/diogorocha/Documents/Plots/timeline.png', 'png')
end
function output = cml_plot(x,y1,y2,y3)
    hold on
    plot(x,y1,color='black', LineWidth=1.5)
    plot(x,y2,color='blue', LineWidth=1.5)
```

```
    plot(x,y3,color='red', LineWidth=1.5)
    xlabel('Time (min)', 'FontSize', 14)
    ylabel('Rewards', 'FontSize', 14)
    title('Cumulative Reward', 'FontSize', 16)
    legend('Rewards','Correct','Incorrect','FontSize',12,'Location','Southeast')
    set(gca, 'lineWidth', 2, 'FontSize', 12)
    xlim([0 x(end)])
    hold off
    saveas(gcf, '/Users/diogorocha/Documents/Plots/cumulative.png', 'png')
end
function output = cml_plot0(x,y1,y2)
    hold on
    plot(x,y1,color='black', LineWidth=1.5)
    plot(x,y2,color='blue', LineWidth=1.5)
    xlabel('Time (min)', 'FontSize', 14)
    ylabel('Rewards', 'FontSize', 14)
    title('Cumulative Reward', 'FontSize', 16)
    legend('Rewards','Correct','FontSize',12,'Location','Southeast')
    set(gca, 'lineWidth', 2, 'FontSize', 12)
    xlim([0 x(end)])
    hold off
    saveas(gcf, '/Users/diogorocha/Documents/Plots/cumulative.png', 'png')
end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%Trial outcome recognition%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function trial_res = outcome_rec(evs)
    comp_data = struct('time_us', {}, 'data', {}, 'names', {});
    for upd_index = 1:length(evs)
        if ~isempty(evs(upd_index).data)
            if ~strcmp('presentation_marker', evs(upd_index).data{end}.name)
                comp_data(end+1).time_us = timeconv(evs(upd_index).time_us-evs(1).time_us);
                comp_data(end).data = evs(upd_index).data{end};
                comp_data(end).names = evs(upd_index).data{end}.name;
            end
        end
    end
    trial_res = struct('Start_Time', {}, 'Outcome', {});
    for comp_index = 1:length(comp_data)
        %this next line corrects the problem where we have more trials than
        %responses because we stopped the game after a trial began
        if strcmp('target_square', comp_data(comp_index).names) && comp_index <
length(comp_data)
            trial_res(end+1).Start_Time = comp_data(comp_index).time_us;
            if strcmp('good_answer', comp_data(comp_index + 1).names)
                if strcmp('reward', comp_data(comp_index + 2).names)
                    trial_res(end).Outcome = "Good/R";
                else
                    trial_res(end).Outcome = "Good";
                end
```

```
            end
            if strcmp('bad_answer', comp_data(comp_index + 1).names)
               trial_res(end).Outcome = "Bad";
            end
            if strcmp('ignored', comp_data(comp_index + 1).names)
                trial_res(end).Outcome = "Ignored";
            end
        end
    end
end

%%%%%%
%this function just converts microseconds into milliseconds and rounds up
%the result so that its easier to plot
function output = timeconv(x)
    output = floor(x/1000);
end
%this function converts milliseconds into minutes
function output = timeconv_min(x)
    output = (x/1000)/60;
end
```