READ ME (version 24102019)

OpenSesame 3.2.8 implementation of the Contralateral Visual Working Memory (CVWM) task in an EEG/eye-tracking setting

Contact

intanwardhani@hotmail.com for further questions about this implementation.

Good to know

If anything is still fuzzy, contact me directly (via email or drop by my office). This task is made to work with two different systems:

1. **SMI RED250 Mobile** eye-tracker (ET) and **actiChamp** EEG system
2. **EyeLink 1000 Plus** ET and **Biosemi** EEG system

Connections:

a. to the SMI ET is elaborated below
b. to the EyeLink ET is not included here yet, but it will be updated soon!
c. with the EEG is pretty straightforward. Functions for registering EEG triggers in both systems are attached separately in this project (check the functions_for_triggers file)

Development with a response box (either Cedrus RB-730 or MilliKey MH-5) is in progress.

Technicalities

<u>Both systems</u>

Make sure you have the right PC resolution setting in the task so that the experiment is presented properly.

In the experimental phase, an adaptive staircase procedure is implemented. Please inspect http://www.psychopy.org/api/data.html#questhandler for further details.

<u>SMI RED250 Mobile and actiCHamp</u>

Triggers starting with code A are sent to the EEG and B to the ET, assuming a serial-to-LPT system. It is also possible (and often better) to send triggers/messages to the ET via Ethernet using the **eyetracker.send_message()** command. Messages sent via Ethernet to the ET will be stored in the idf file as an MSG type sample.

Another possibility is to log a message using the **eyetracker.log()** function. The message will be logged by PyGaze in the SMILOG file. This could be useful for double-checking if all triggers or messages are registered.

EyeLink 1000 Plus and Biosemi

A direct parallel port is available in this system. Triggers to the EEG are sent as integers. Triggers to the ET are sent via Ethernet using the **eyetracker.send_message()** command. Messages sent via Ethernet to the ET will be stored in the idf file as an MSG type sample.

Another possibility is to log a message using the **eyetracker.log()** function. The message will be logged by PyGaze in the SMILOG file. This could be useful for double-checking if all triggers or messages are registered.

Device settings

SMI RED250 Mobile and actiCHamp

In our lab, there are 3 computers + 1 presentation monitor. The presentation monitor is located in the testing room only to present stimuli to participants. The other 3 computers are for the experimenter to control the experiment. One computer for EEG acquisition, one for controlling the ET, and the last one as a stimulus computer (i.e., local PC). EEG data is saved in the EEG computer and the only connection between the local and EEG PCs is via the serial-to-LPT box to send triggers. The stimulus and ET PCs are connected via Ethernet (on top of serial-to-LPT box), hence it is possible to send richer information and a more complicated command to the ET PC.

When using OpenSesame to generate the experiment, we are required to do some tweaking. The reason is that we have a dual-pc setup, yet PyGaze (the Python library that controls the ET from within OpenSesame) assumes a single-pc setup. So, there are two main things we have to do:
1. Define the IP address of the ET computer and another of the local computer. The addresses and send/receive port are described in our lab documentation.
2. Customise the file path for storing the idf files that are generated by the ET. Make sure you have created a folder for your experimental data.

This tweaking is done by changing some lines of codes in the **libsmi.py** library and going back and forth to your OpenSesame experiment builder.

IMPORTANT!
You cannot change the SMI library in the OpenSesame program that is installed in the computer (because you do not have the administrator rights). What you can do is to install OpenSesame as a zip folder and extract it in your own folder. Then you can change the SMI library that is contained in your own folder.

Please follow the steps below on how to modify your libsmi.py.
1. Line 127

By looking at the default IP address, we can see that PyGaze assumes a single-pc setup. We can change the value of the iViewX IP address directly here. It is also important to adjust the values for send port and receive port, if applicable.

2. Line 208
The values from line 127 are used on line 208.

```
res = iViewXAPI.iV_Connect(c_char_p(ip), c_int(sendport), c_char_p(ip),
c_int(receiveport))
```

The first **ip** is the iViewX IP address, then **send port**, then the second **ip** that corresponds with the IP address of the local PC, and lastly the **receive port**. Just directly customise the second "ip" on this line as a string.

3. Line 128
There is a **logfile** argument that determines the path where we store the data. A connection between OpenSesame and SMI will result in 3 outputs: The OpenSesame csv file, the SMILOG txt file, and the SMI idf file (i.e., the eye-recording data). The csv and txt files are stored automatically in the same path (folder) as the experiment in the local PC. This is what the

```
logfile=settings.LOGFILE
```

does: obtaining the full path of the current experiment folder. We can set this up via the **pygaze_init** item on OpenSesame. The "logfile" argument is used on line 194 to generate the SMILOG txt file. (You don't need to change anything on line 194).

4. Line 167
By looking at the value of **self.outputfile**, we can see again that PyGaze assumes a single-pc setup. If we use the same path as for the SMILOG file, it will not be able to save the idf file. So, we need to customise line 167. For example, I customise it in the following way:

```
self.outputfile = "C:/Users/iViewX/Desktop/logfiles/Intan/" +
logfile[35::] + ".idf"
```

The first bit is the path to the log folder on the SMI eyetracker (in string). The second bit is how I name each file. I want to name it the same way as I name the OpenSesame and SMILOG files. The last bit is the format of the file (this is important, otherwise your eye data file can't be generated properly).

5. Lines 168 and 169
Basically, the purpose of changing these lines is the same as changing Line 167: to create a meaningful data file. By modifying Line 168, you can save the name of your experiment in the idf file. By modifying Line 169, you can save your participant code name in the idf file. (I think all of these are pretty self-explanatory).

6. Adding a new function
The original libsmi.py library does not contain a **send_message()** function. If you want to send triggers to the ET via Ethernet, you need to add this function yourself. Put the following code on Line 695 under the send_command() function. Essentially, you can put it anywhere you like, but I think this is the location that makes more sense.

```
def send_message(self, msg):

        """Sends a message to the eye tracker. Useful for sending timestamped markers directly
        to iViewX.

        arguments
        msg     -- the message (a string value) to be sent to iViewX

        returns
        Nothing
        """

        try:
            iViewXAPI.iV_SendImageMessage(c_char_p(msg))
            # print message to debug window is optional:
            print("message: ", msg)
        except:
            raise Exception("Error in libsmi.SMItracker.send_message: failed to send markers to iViewX")
```

It is important to know that a downside of doing the tweaking above is that you have to modify Lines 167-169 when you run different experiments. Repeat: **You always have to remember to modify every time you are about to run your experiment!** (An extra task that can also tweak your brain).

About the task

The task is called Contralateral Visual Working Memory task because the targets to be memorised are presented contralateral to the distractors.

Participants are presented with either one or three targets on either side of the fixation point at the centre of the screen. Targets are always presented with distractors. The targets and distractors are in the shape of Pacman- and Tetris-like pieces, depending on the participant code name (Pacman-like for even numbers and Tetris-like for odd numbers). After a retention interval, a target is presented alone and participants have to indicate if there is a feature change in the target. Participants have to make a response as accurately and quickly as possible by pressing a designated key. If you would like to change the keys, you can go to the **correct_response item** and adjust the conditional statement.

About the paradigm

This paradigm is used to test the contextual effect of reward on motivation and cognitive control.

The experiment consists of two blocks, namely Reward (R) and No Reward (NR) blocks. The R block contains incentivised (R_r) and non-incentivised (R_nr) trials.

The NR block contains only non-incentivised trials (NR_nr). The number of NR_nr trials in the NR blocks are the same as the number of R_nr trials in the R blocks. Participants are informed beforehand whether the upcoming block is an R or NR block.

At the beginning of every trial, participants learn whether the current trial comes with a reward or not. This is cued by a triangle (for odd-numbered participants) or a diamond (for even-numbered participants) preceding every trial. If a participant is reward-cued with a triangle, then a diamond is a cue for no reward (and vice versa). At the end of each trial, participants receive feedback about their performance as the following.
"+1" → fast and correct responses in R_r trials → rewarded
"Next trial" → fast and correct responses in R_nr and NR_nr trials → not rewarded
"- -" → incorrect or slow responses → not rewarded