Codes for the virtual hallway

Here are the things that I’ve created/modified to have the code run a virtual hallway:

1. I created 2 patterns: one of them is a dark stripe that expands horizontally as the animal moves forward, and the other one is a little dark stripe that expands both vertically and horizontally as the animal moves forward. The codes that I made for generating those 2 patterns can be found in the folder and are called ‘make\_expanding\_virtual\_hallway’, and ‘make\_expanding\_virtual\_hallway\_2dim’. Those are pretty ugly because I couldn’t figure out a better way of doing them, sorry :s. You’ll have to change the zeros & ones pattern to fit the position of your arena. These are also made for 36 panels because my arena from the behavior room has 3 rows, but you can change it to 24 panels very easily. I have that version for the 2P if you want it.
2. I added an option to the ‘fly\_experiments.mat’ to write the percentage of probe trials. I modified the GUI accordingly, and also added an extra experiment type, called ‘virtualHallway’ to it.
3. In the ‘begin\_trials.mat’ code, I’ve added different parameters for the results figure when the experiment is a virtual hallway. In that code, I’ve also added some variables to store multi-trial data, and I’ve defined the percentage of probe trials and sampled them randomly from the total number of trials. I’m also sampling 3 trials to be empty trials. If we were interested in varying this number, an extra option could be added to the GUI, but for now I didn’t think that was necessary.
4. I’ve made two new codes that are called by ‘begin\_trials.mat’ to run the virtual hallway trials. One of them, ‘run\_trial\_vHallway.mat’ runs the trials for which there is visual stimulation. If runs both the opto trials and the probe trials, and that probe\_status is defined in the ‘begin\_trials.mat’ code. The main different with the pre-existing ‘run\_trial.mat’ is that the NiDaq signals are acquired in the background so that the acquisition time doesn’t have to be pre-set. Instead, I run them in the background and added a listener that calls to the function ‘stopF.mat’ (also in the folder) that acquires the data until the voltage in channel 5 (in my case the channel that’s receiving the data from the y dimension of the panels) reaches a certain voltage that indicates the stimulus has reached its maximum expansion. I’m going to add a 2 sec pause inside that function so that the acquisition stops 2 sec later and that we can then get some data on what the animal is doing once the opto pulse has ended. A second listener inside this code calls the function ‘logDaqData.mat’, which saves the acquired data. The additional functions ‘loadFromLogFile.mat’ and ‘loadTimeFromLogFile.mat’ are used to log and store the data and time data in the appropriate format.

The other code that’s called is ‘run\_empty\_trial.mat’, which runs a 10 sec trial with foreground acquisition but no visual stimulation.

1. I’ve made a new code that is called by ‘begin\_trials.mat’ to display the results figure. It is called ‘display\_trial\_vHallway.mat’. Instead of the previous figures, this code plots:

-the y dimension of the stimulus as a function of time, which is analogous to showing the position of the animal down the hallway.

-the forward velocity as a function of time.

-the yaw speed as a function of time.

-a histogram of forward velocity for every trial.

-a histogram of yaw velocity for every trial.

-a raster plot of the forward velocity as a function of time for every trial.

-a raster plot of the angular speed as a function of time.

-a scatter plot of the trial duration vs the trial number, distinguishing between probe, empty, and opto trials.

I’m also overlaying the opto trigger channel on top of the first three plots, and I’ve added a text on top to say if it is currently a probe or an opto trial. There are some things to be improved in these plots, that I’m currently working on: I would like to see the different between probe and opto trials for the velocities but right now I’m not differenciating those two in the raster plots. Because there are so many trials (I’ve run 100 trial sessions) it is very hard to try to cram too many things there, and I think plotting the mean would hide the effect of what type of trial came before the current one and how was the animal behaving then. To overcome this, I’m trying to do these plots in the analysis code instead. It would also be nice not to have to downsample or upsample the velocities, and just plot them as a function of distance instead. It sometimes happens however that the animals take little movements backwards, so plotting the velocity as a function of distance gives you weird looking points if you use lines. You can bypass this using just dots, but too many trials overlaid look confusing. I’m working on this.

1. In terms of the python code, I’ve changed some things in the ‘fictrac\_2d.py’ code. I’ve added ‘self.voltage\_out’ and ‘self.final\_time’. The first one is defined in the code ‘analogout.py’. When called, it returns the value of the voltage output from the x gain channel (which is what I’m using to move the visual stimulus, and what I want to use to trigger the LED stimulation). The second one is a time that will later be set to be equal to the self.time\_elapsed when the desired voltage has been reached. When the experiment type is the virtual hallway and these two times are equal, the LED pulse is triggered. For the LED to be triggered properly, I’ve also modified the ‘protocol.py’ code, adding the function ‘virtualHallway(self,voltage\_out)’, that sends a pulse of value 7 when the voltage from the x gain channel is in the desired range. I made two extra codes ‘analogout\_empty.py’ and ‘run\_aout\_empty.py’ to run the empty trials. This could probably be done in a better way by adding an extra parameter to the regular code, but I didn’t figure it out shortly and didn’t want to waste time doing this.
2. I’ve added a post-hoc analysis code to the folder as well, called ‘virtualHallwayAnalysis.mat’, as well as auxiliary analysis functions, inside a folder called ‘analysis’. ‘singleTrialVeloicityAnalysis9mm’ filters and smooths fictrac’s data (it’s an alternative to Yvette’s fictracsignaldecoding’, and the one I usually use. ‘loadSettings’ loads the settings of the NiDaq channels in the different rooms.

Some things to be aware of:

1. I’ve changed the NiDaq settings to match my setups in the ‘process\_data’ file, and maybe somewhere else if it is defined elsewhere?, so that will need to be changed.
2. I’ve changed the analogout.py code to match my phidgets channels, but also to output y instead of yaw. I only use yaw gain, but I save the side motion instead. This will need to be changed as well if you want it differently.
3. I’ve changed the starting frames in the process\_panel\_360.m, so you don’t want to copy that one.

Some pending issues:

1. The empty trial start is triggered earlier in time than the start of the 2 other trial types. That shouldn’t be a problem because there is no visual stimulus for the animal to associate with it, but when looking at populational data, it has to be aligned (I’ll add something to the analysis code to do this).
2. I’m now running both the probe and the opto trials with the same code, and calling the Arduino in both cases but not setting the pulse to have a value >0 in the probe trials so overall, they seem to all start at the same time. It appears as though there might be some very small variations between all the trials (including within the same category). These differences seem to be at most of about 0.4 s so I think it should be ok. I also think it probably can’t be improved, but we should align the data to the start of the python code.
3. Because of how the hallway is coded, once the pulse is triggered, the code sleeps for the duration of the pulse (0.5 s) and then breaks out of the acquisition loop and stops. I think it would be nice to acquire the animal’s behavior during and a little after the opto pulse, but I can’t do it with the current code config.