Help Manual

System requirements:

Hardware:

Balltrack setup with two optical mouse sensors for reading data and a working network connection between the computer with the visual stimuli software and the one with FlyTracker. Network connection is not mandatory barring for using network trigger.

Software:

- Ubuntu Linux (for sure works on: 10.04 LTS and 12.04 LTS, no other tested)
- Matlab (for sure works on: R2012a no other tested)

Setup:

Put the FlyTracker-folder anywhere on the desktop but make sure that a path (with subfolders) is added to it in matlab.

There is a lot of reading and writing to files going on in FlyTracker, relative paths are set within in the FlyTracker folder and therefore it is very important that files are not moved manually within in the FlyTracker folder. Only exception is data from experiments as these are never read from, you can also set the save path to anything you like in the GUI.

Configuration

Calibrating the setup...

Trigger

No Trigger

System can be run without trigger, either by having the user manually start and stop the recording or setting a timer for how long it should record.

Setting up the network trigger client

The simple trigger client is implemented in python and is provided as the file TriggerClient.py. This file needs to be called from the visual stimuli software so that commands can be sent over the network (using TCP as transport layer protocol).

Currently the Python client supports communication via a named pipe, this is a sufficiently fast solution for interprocess communication that is also allowing many different platforms for the visual stimuli implementation.

The server accept three types of commands, “start”, “pause” and “quit”. Needless to say you first need to start the recording by sending “start” and finally its important that the server is stopped by providing the “quit” command as otherwise it will run until time out.

To be able to run it three things needs to be configured.

Host: Set this to IP-address or hostname of FlyTracker, the information can be found in the menu of the main window of FlyTracker. Choose Settings>Network Settings.

Use either hostname or IP-address, the former is a better choice as IP-address is dynamically generated at startup and therefore would need to be changed everytime the computer with FlyTracker is restarted.

Port: Use the same port number in the client as in FlyTracker. This can be changed but beware that the new port is open and valid. It normally needs to be larger than 1024 and lower than 65534.

Pipe: the pipe in TriggerClient.py needs to have the same filename and path as the pipe created in the visual stimuli software.

Matlab sample code for the stimuli system:

Before drawing commences:

```
pipe = '/home/Username/pipe';
system('mkfifo /home/Username/pipe');
system('python /home/..../TriggerClient.py');

fid = fopen(pipe,'w');
fwrite(fid,'start');
fclose(fid);
```

Pause:

Pauses can be written to the pipe the same way with 'pause' instead of 'start'. Exactly how pauses are set depends on implementation of the visual stimuli software.

After drawing is done:

```
fid = fopen(pipe,'w');
fwrite(fid,'quit');
fclose(fid);

system('rm -f pipe');
```

Plotting

Saving data

Data is saved in different ways depending on what kind of trigger system is used. The recording has three states, “started”, “paused” and “stopped”. One datafile is created from the moment when it first is started until its stopped. If the recording is paused during a recording that will not create a new data file but rather divide the file into data blocks.

Currently only the network trigger supports pauses.

Data is saved into .mat-files with the filename of the current time with second precision. Each data file consists of one 4xn cell where n is the number of blocks. The elements of the first row in the cell consists of arrays of forward velocity, second row is for sideway velocity, third is for rotational velocity and finally the forth one is a time stamp. Each column represents data for that particular time stamp.

The Python-Matlab interface

The main application uses both Matlab and python code. The communication between them is limited as much as possible and is exclusively done via system calls reading from and writing to named pipes.

A possible extension to the current GUI would be to implement pseudo real time plotting using pipes at the same time as data is written to it. Right now data is continuously written to a temporary data file that is then parsed and saved correctly by the Matlab side of the code.

Matlab package

MVC pattern isnt really implemented, basically only seprating GUI-files from the rest of the files.

Model
w

View
w

Python package

DAQ file

MouseHandler
This class deals with instantiating the mice, inherits from threading.Thread and runs a loop that evaluate the mice coordinates and saves them to file anytime they read data and also timestamps this.

Mouse class tree
Currently two types of mice, SensorMouse and IdMouse but TriggerMouse could easily be added. Inherits from AbstractMouse which in turn inherits from threading.Thread.

Utilities

A utility class providing static methods for different kinds of data manipulation, file I/O and exception logging for debugging.

Known bugs:

- Currently “no-trigger” recording only works if its the first recording you do. Code blocks at fopen(pipe) in this case. Prio: medium

FAQ