#=======================================

#=======================================

# Eyelink 1000 Documentation

#=======================================

#=======================================

Written by Jonathan van Leeuwen, 2017

#=======================================

# Using the parser

#=======================================

###

# Using the parser, tl;dr

1: Convert the .edf file(s) to .asc

files using the eyelink converter.

2: Select the .asc file(s)

3: Make sure that the start/stop key,

variable prefix and parser settings

are correct.

4: Click on "Parse" and wait until the

parser is finished.

5: Parsed files are saved in the same

directory as the .asc file(s).

6: Done!

###

# Inputs:

Start trial key

Default = start\_trial

This is the string it looks for to

determine were a trial starts and

therefore were to start looking for

relevant data

Stop trial key

Default = stop\_trial

This is the string it looks for to

determine were a trial stops and

therefore were to stop looking for

relevant data

Variable prefix

Default = var

This is the string it looks for to

determine what are experiment variables

See settings for a more detailed explanation

of how the data is extracted.

###

# Caveats

As there are a multitude of ways to log the data

and each one having its own specific configuration

there is a possibility that the parser wont be able to parse the data.

For instance, the parser expects to get data from

the regular expressions in a specific format, if the format

does not match then the merging and data calculation probably

does not work.

That being said.

If that is the case then some slight modifications to the parser

function should do the trick:

remove/add the relevant keywords

Double check the new indices finding keywords (you don't want timestamps used for curvature calculation)

Remove any data calculations not used, e.g. curvature calculation

The parser was written and tested using

Anaconda python 2.7 - 64bit.

#=======================================

# Dependencies (required libraries)

#=======================================

re

os

sys

time

math

numpy

PyQt5

pandas

psutil

itertools

collections

multiprocessing

###

# Installing dependencies

If you use Anaconda python 2.7 64bit (recommended) most

of the libraries are already installed.

The remaining libraries can be installed

by either method 1 or 2 or 3.

Method 1 - Use this method if you are using Anaconda

Open Anaconda promt

conda install library\_name

Example: "conda install pandas"

Method 2 - dont use "pip" if you are using Anaconda

Open a console and type:

pip install library\_name

Example: "pip install pandas"

Method 3

If method 1 does not work you can go

to "http://www.lfd.uci.edu/~gohlke/pythonlibs/".

Download the .whl file.

Open a console and type:

pip install "fileDirectory\fileName.whl"

Example: pip install "D:\Work\filename.whl"

###

# When using Anaconda 64it python 2.7 (Recomended)

After succesful instalation of Anaconda python 2.7 – 64bit,

use the recommended editor “Spyder 3”.

Before running the “EyeParser.py” script you need to

change some settings in spyder 3.

Tools -> Preferences -> IPython console -> Graphics --- Change backend to “Automatic”

#=======================================

# Settings

#=======================================

###

# Regular expressions

# Samples

Default = r"(\d{3,12})\t\s+(\d+\..)\t\s+(\d+\..)\t\s+(\d+\..).+\n"

With the default expression input data should look like:

50475 812.4 492.2 2022.0 ...\n

Without changing the source code outputs should

have the following format:

The first data extract saves the timestamp: (\d{3,12}) -> "50475"

The second data extract saves the xPosition: (\d+\..) -> "812.4"

The third data extract saves the yPosition: (\d+\..) -> "492.2"

The fourth data extract saves the pupil size: (\d+\..) -> "2022.0"

# End fixation

Default = r"EFIX\s+[LR]\s+(\d+)\t(\d+)\t(\d+)\t\s+(.+)\t\s+(.+)\t\s+(\d+)\n"

With the default expression, input data should look like:

EFIX L 49826 50555 730 813.3 491.8 1991\n

Without changing the source code outputs should

have the following format:

The first data extract saves the start of fixation: (\d+) -> "49826"

The second data extract saves the end of fixation: (\d+) -> "50555"

The third data extract saves the fixation duration: (\d+) -> "730"

The fourth data extract saves the x fixation position: (.+) -> "813.3"

The fifth data extract saves the y fixation position: (.+) -> "491.8"

The sixth data extract saves the pupil size duringfixation: (\d+) -> "1991"

# End saccade

Default = r"ESACC\s+[LR]\s+(\d+)\t(\d+)\t(\d+)\t\s+(\d+.?\d+)\t\s+(\d+.?\d+)\t\s+(\d+.?\d+)\t\s+(\d+.?\d+)\t\s+(\d+.?\d+)\t\s+(\d+)\n"

With the default expression, input data should look like:

ESACC L 50556 50563 8 813.0 489.0 830.6 490.0 0.36 49\n

Without changing the source code outputs should

have the following format:

The first data extract saves the start of saccade: (\d+) -> "50556"

The second data extract saves the end of saccade: (\d+) -> "50563"

The third data extract saves the saccade duration: (\d+) -> "8"

The fourth data extract saves the x start saccade position: (\d+.?\d+) -> "813.0"

The fifth data extract saves the y start saccade position: (\d+.?\d+) -> "489.0"

The sixth data extract saves the x end saccade position: (\d+.?\d+) -> "830.6"

The seventh data extract saves y end saccade position: (\d+.?\d+) -> "490.0"

The eight data extract saves the saccade amplitude: (\d+.?\d+) -> "0.36"

The ninth data extract saves the saccade peak velocity: (\d+) -> "49"

# End blink

Default = r"EBLINK\s+[LR]\s+(\d+)\t(\d+)\t(\d+)\n"

With the default expression input data should look like:

EBLINK L 51117 51358 242\n

Without changing the source code outputs should

have the following format:

The first data extract saves the start of the eye blink: (\d+) -> "51117"

The second data extract saves the end of the eye blink: (\d+) -> "51358"

The third data extract saves the eye blink duration: (\d+) -> "242"

# Start trial

Default = 'MSG\\t(\d+)\s+('+startTrial+').\*\\n'

startTrial = The string input from "Start trial key"

With the default expression input data should look like:

MSG 55095 start\_trial\n

Without changing the source code outputs should

have the following format:

The first data extract saves the start of the trial: (\d+) -> "55095"

The second data extract saves the keyword used to define trial start: ('+startTrial+') -> "start\_trial"

# Stop trial

Default = 'MSG\\t(\d+)\s+('+stopTrial+').\*\\n'

stopTrial = The string input from "Stop trial key"

With the default expression, input data should look like:

MSG 55095 stop\_trial\n

Without changing the source code outputs should

have the following format:

The first data extract saves the end of the trial: (\d+) -> "55095"

The second data extract saves the keyword used to define trial end: ('+stopTrial+') -> "stop\_trial"

# Variables

Default = 'MSG\\t(\d+)\s+('+var+')\s+(.+)[\s+]?.\*\\n'

var = The string input from "Variable prefix"

With the default expression, input data should look like:

MSG 58956 var correct\_response 1\n

Without changing the source code outputs should

have the following format:

The first data extract saves the time the variable was sent: (\d+) -> "58956"

The second data extract saves the variable prefix: ('+var+') -> "var"

The third data extract saves the variable name: (.+) -> "correct\_response"

The fourth data extract saves whatever was sent after the variable name: [\s+] -> "1"

# Other messages

Default = 'MSG\\t(\d+)\s+(?!'+var+'|'+startTrial+'|'+stopTrial+')(.+)[\s+]?.\*\\n'

var = The string input from "Variable prefix"

startTrial = The string input from "Start trial key

stopTrial = The string input from "Stop trial key"

This saves all messages which are not start, stop or variable messages.

Used as a catch all.

With the default expression, input data should look like:

MSG 63902 badLogging some data

Without changing the source code outputs should

have the following format:

The first data extract saves the time the data was sent: (\d+) -> "63902"

The second data extract saves everything after the timestamp: (.+) -> "badLogging some data"

###

# Various settings

Parsed name

Default = Parsed

This is the string which is appended to the

parsed file name before saving

Raw name

Default = Raw

This is the string which is appended to

the raw file name. Its appended after

the parsed name.

(parsed name + raw name)

Merged name

Default = AllDataMerged

This is the string appended to the

file containing the merged data sets.

Merge files

Default = No

Whether or not to merge the files you

are parsing. Only works if the number

of files you are parsing is larger

than 2.

Save raw file

Default = No

Whether or not to save the raw file.

The raw file only contains:

Time stamps.

X gaze coordinate.

Y gaze coordinate.

pupil size.

Parallel processing:

Defaults to No if less than 2 files

Defaults to Yes if more than 1 files

Automatically sets itself, but can be over

written.

CPU cores

Default = 64

Defaults to the max number of threads in the system -1

The number of jobs created for parsing the data.

You should not exceed the number of files.

If you use the same number as you have cores/thread

the system might become unstable while parsing.

Pixels per degree

Default = Automatic

Determines whether pixels per visual degree is calculated

based on saccade amplitudes and positions (automatic)

or manual preset value.

Pixels per degree is used for some of the calculations.

Which samples to use for saccade curvature calculation.

Px per deg(manual)

Default = 48

How many pixels on the screen correspond to 1 visual degree.

Only used if Pixels per degree is set to "Manual"

#=======================================

# Logging data

#=======================================

Preferably you start a trial by sending

a start trial keyword:

"start\_trial"

then you send all variables with 2 ms

between each variable:

"var condition A"

wait 2 ms

"var trialNr 10"

ending the trial with a stop trial keyword:

"stop\_trial"

The parser epochs the data around the

start and stop keys, and logs all variables

and data between these two keys.

Any variables sent before the start trial key

or after the stop trial key will not be logged.

For example:

"This data is ignored"

MSG 45845 start\_trial

"All data between 45845 and

50484 will be saved"

MSG 50484 stop\_trial

"This data is ignored"

When logging data try to only send

1 value or string with each message.

Don't send the same variable multiple times

during one trial.

Do:

"start\_trial"

2ms lag

"var condition a"

2ms lag

"var trialNr 11"

2ms lag

"var distrXpos 250"

2ms lag

"var distrYpos 100"

2ms lag

run experiment

"stop\_trial"

Don't:

"var trialNr 11"

"start\_trial"

"start\_trial"

"distrXYpos 250 100"

"distrXYpos 250 100"

run experiment

"stop\_trial"

"var condition a"

#=======================================

# Parsing pipeline

#=======================================

1: Constructs regular expressions.

2: Constructs keywords data frames.

3: Opens and reads file.

4: Extracts data using regular expressions.

5: Puts all the data into pandas data frames.

6: Creates dicts to store temporary data.

7: Iterates through each trial based

on start/stop times

8: For each trial extracts all eyetracking

events, variables and messages

9: For each trial extracts eye position,

speed and pupil size for all saccades

and fixations

10: For each trial calculates the speed of

eye movements

11: For each trial calculates the curvature

of each saccade.

12: Combines all the data into one data frame

13: Saves the data frame as a pickle (.p) file

#=======================================

# Output data

#=======================================

The following keywords are written to the

data frame, all keywords are preceded by

the prefix "DK\_".

All user sent messages are preceded with

the prefix "DV\_".

###

# Data keywords and what is stored for each row

All eye position related values are in pixels

unless otherwise specified

DK\_trialNr = Trial number (float)

DK\_sTrial = Timestamp for trial start (float)

DK\_eTrial = Timestamp for trial end (float)

DK\_sMsg = Message used for detecting start of trial

DK\_eMsg = Message used for detecting end of trial

DK\_esacc = list of times for end of saccades (list of floats)

DK\_ssacc = list of times for start of saccades (list of floats)

DK\_durSacc = list of saccade durations (list of floats)

DK\_ssaccX = list of start saccade x positions (list of floats)

DK\_ssaccY = list of start saccade y positions (list of floats)

DK\_esaccX = list of end saccade x positions (list of floats)

DK\_esaccY = list of start saccade Y positions (list of floats)

DK\_saccAmp = list of saccade amplitudes in visual degrees (list of floats)

DK\_peakVelocity = list of peak velocities of saccades in degrees/s

DK\_saccTraceX = list of lists of all X positions during all saccades (list of list of floats)

DK\_saccTraceY = list of lists of all Y positions during all saccades (list of list of floats)

DK\_saccTracePup = list of lists of all pupil sizes during saccades (list of list of floats)

DK\_saccTraceTime = list of lists of all sample times during all saccades (list of list of floats)

DK\_sFix = list of times for start of fixations (list of floats)

DK\_eFix = list of times for end of fixations (list of floats)

DK\_durFix = list of times for fixation durations (list of floats)

DK\_fixX = list of start fixation x positions (list of floats)

DK\_fixY = list of start fixation Y positions (list of floats)

DK\_fixPup = list of average pupil sizes during fixation (list of floats)

DK\_sBlink = list of blink onset times (list of floats)

DK\_eBlink = list of blink end times (list of floats)

DK\_durBlink = list of blink durations (list of floats)

DK\_rawX = list of all gaze x positions during the trial (list of floats)

DK\_rawY = list of all gaze Y positions during the trial (list of floats)

DK\_rawTime = list of all gaze times during the trial (list of floats)

DK\_rawPupSize = list of all pupil sizes during the trial (list of floats)

DK\_euclidDist = list of Euclidian distances between all samples in a trial (list of floats)

DK\_curvature = list of median saccade curvatures for all saccades (list of floats)

DK\_fixTraceX = list of lists of all X positions during all fixations (list of list of floats)

DK\_fixTraceY = list of lists of all Y positions during all fixations (list of list of floats)

DK\_fixTracePup = list of lists of all pupil sizes during fixations (list of list of floats)

DK\_fixTraceTime = list of lists of all sample times during all fixations (list of list of floats)

DK\_saccAngle = List of saccade angles, between -180 and 180

DK\_includedTrial = Bool - Defaults to all values as True, Use this for filtering data later (integrated with plotter)

DK\_pixPerDegree = Float - The calculated pixels per degree based on saccade amplitude (might deviate slightly from expected value)

DK\_RMSDeg = list of floats - The root mean square noise for each fixation in visual degrees

DK\_RMSPix = list of floats - The root mean square noise for each fixation in pixels

DK\_stdvDeg = list of floats - The sdtv noise for each fixation in visual degrees

DK\_stdvPix = list of floats - The sdtv noise for each fixation in pixels

###

# Variables written to data file

# preceded with the variable keyword

The first string after the keyword

is saved as the column name while

the value directly after the first space

is written for each trial (row)

The time the message was sent is

put in a different column with the

same name + TimeStamp

Example data line parsing:

MSG 1681525 var Ydistr 767

The preceding line will be saved

as two columns in the data frame:

DV\_Ydistr (column) with the value 767 (row)

DV\_YdistrTimeStamp (column) with the value 1681525 (row)